Improving In-game Gesture Learning with Visual Feedback

Matthias Schwaller[1], Jan Kühni[1], Leonardo Angelini[2,1], and Denis Lalanne[1]

[1] Department of Informatics, University of Fribourg, 1700 Fribourg, Switzerland
{Matthias.Schwaller,Jan.Kuehni,Denis.Lalanne}@unifr.ch
[2] University of Applied Sciences and Arts of Western Switzerland, Fribourg
Leonardo.Angelini@hes-so.ch

Abstract. This paper presents a research work on gesture recognition and feedback to reduce the learning time of new gestures and to augment user performance in a game application. A Wiimote controlled space shooter game, GeStar Wars, has been developed. The player controls a spaceship through the buttons in the controller, while forearm gestures can be used to perform special actions. Gesture strokes are mapped in a 3x3 grid and are differentiated according to the path of the covered grid cells. In-game visual feedback displays to the user the current gesture path and which cells were covered after the gesture is performed. The novelty of this research resides in the correlated gesture recognition methodology and feedback which helps the user to learn and correct the gestures. The evaluation, conducted with 12 users, showed that the users performed significantly better if feedback was provided.

Keywords: Gestural interfaces, User evaluation, In-game Feedback, Accelerometer.

1 Introduction

Gesture interfaces became very popular for human-computer interaction. Gestures are very practical in some application scenarios, for instance, for entering a command or to mimic a human movement for an avatar. Although most of the gestures are somehow natural and intuitive to use, learning is still required in an initial phase, because metaphors and icons are not always sufficient to describe the gestures. Learning, however, is time consuming; therefore, some users may prefer an alternative interface if this phase takes too much time. This is a typical issue of gestural interfaces that we want to cope with in this research work. To this purpose, in this paper we explore the advantages of having visual feedback that helps the users to see which movements they have to do and where they did errors, in case they did. In today's systems, there is sometimes no additional feedback besides the action that was performed; indeed, when the command is not executed, often the user cannot understand why the system did not recognize the gesture.

Games are a very specific application domain for gestures. User skills have a very important role in games and the development of those skills is one of the most important benefits that videogames can introduce. Subrahmanyam and Greenfield showed for example that videogames can increase spatial skills for both girls and boys [1]. The Nintendo Wii, thanks to the Wii Remote (or Wiimote), introduced a new dimension for the skills that can be developed by the user: through forearm gestures and body movements the user can improve skills that were completely ignored in classic video-games. However, a big challenge for game developers is that different people perform the same gesture in several different ways. In order to recognize gestures two strategies can be adopted:

1) the system is trained and adapted for each user with his own gestures,
2) the user learns how to correctly perform gestures.

In both cases additional time is required for the user. While a User Centered Design approach suggests designing simple gestures that can be easily learned by every user, gaming scenarios can benefit of more difficult gestures as part of the game challenge. Indeed, GeStar Wars, the game we present in this paper, adopts Wiimote gestures that are not very easy to perform for a novice user, although they can be learned with proper training.

In GeStar Wars, we implemented visual feedback that helps the user to show how the gestures are interpreted. Feedback is composed of two levels. The first level shows the smoothed accelerometer values in a snake-like visualization. The second level shows a 3x3 grid as soon as the user finishes a gesture. The grid permits to see what was wrong in a gesture, by showing which cells of the grid were covered by the gesture path and which cells were missed.

To evaluate the impact of gesture feedback, a user evaluation with 12 users was conducted. With the game we are able to measure the performed gestures that were successfully (with and without errors) and those that were not successfully. Further the game permits also to compare the scores.

The remainder of the paper is structured as follows: first a literature overview about gesture feedback is presented; then, the game application is explained. Furthermore, the gestures and feedback are illustrated. Finally, an evaluation with its results is presented, followed by conclusion and future work.

2 Related Work

Forearm gestures are very popular in many application domains and several accelerometer based algorithms exist for their recognition [2–5]. However, feedback for stroke-like gestures or metaphoric gestures is still a research topic that needs further investigations. Often, no information is provided on why a gesture was not recognized. This may be frustrating, especially for beginners.

A clever approach to help the user learn single-stroke gestures was introduced by Bau and Mackay with OctoPocus [6]. OctoPocus, combines feedback and feedforward in order to provide hints even while the user is performing the gesture. This system allows the user to know at any time in which direction s/he has to continue in order to fulfill the desired single-stroke gesture.

A more complex feedback approach was introduced for Gesture Play by Bragdon et al. [7]. The goal of Gesture Play is to help and motivate users to learn gestures through an awarding system. In their system both feedback and feedforward is provided. Gesture Play shows labels after the gesture performance in order to inform the user if the gesture was correct or not, and it shows a semi-transparent hand to help placing the hand in the correct position, as well as wheels and springs to illustrate how the action should be performed. In our application we are showing which case of the 3x3 grid has to be passed and in which order.

The GestureBar, introduced by Bragdon et al. [8], supports the users in learning pen-based gestures through a toolbar, a typical widget of many GUI applications. With the GestureBar users do not need any training or prior gesture experience to start using it, since it is always showing to the users which gestures can be performed and how they can be performed.

Li et al. presented in their second experiment a way to help the user to learn the gestures [9]. To get help, the user has to activate a widget that shows each command and the related pen-based gesture. This system allows learning a gesture again each time the user feels unsure about how to perform a command. In our system, feedback is showed immediately either after a not recognized gesture or whenever the user presses and quickly releases the gesture segmentation button.

An on-demand assistance for multi-touch and whole-hand gestures was also presented in ShadowGuides [10]. The authors showed that users remembered more gestures while learning with ShadowGuides than with video-based instructions. In GeStar Wars we introduced a tutorial level to learn the gestures; moreover, a video-tutorial was shown before the game.

Kela et al. [2] presented an accelerometer-based gesture recognizer for metaphoric 3D gestures. The gestures they presented were used to operate a VCR and to operate in the Smart Design Studio where they use gestures such as up, down, left, right, push and pull. To perform a gesture the user has to press a button at the beginning of the gesture and release it at the end. In GeStar Wars we adopted a similar technique for gesture segmentation, although gesture dimensions are limited to the x-z plane.

Several lightweight algorithms have been implemented in order to recognize accelerometer based gestures. Those algorithms require generally no or a very little training set, thus speeding up development and user adoption. A gesture recognizer for accelerometer sensors that can be quickly implemented in prototyping scenarios is presented by Kratz and Rohs [3]. Another accelerometer based gesture recognizer implemented for the Wiimote is presented by Liu et al. [4]. They also segment gestures by pressing the A-button on the Wiimote while the gesture is performed. The same segmentation technique was adopted also by Schlömer et al. [11]. To facilitate the learning phase, they introduced drawings of the gestures and demonstrated the execution of the first gesture. Another Wiimote gesture recognizer based on A-button segmented gestures was presented by Wu et al. [12]. Their system recognizes direction gestures, shape gestures and one-stroke alphabet letters. Their gesture shapes are similar to GeStar Wars one-stroke gestures.

While GeStar Wars segmentation and recognition techniques are similar to previous systems, our recognition algorithm is meant to facilitate the user understanding of the system and learning phase. Indeed, provided feedback reflects the behavior of the system and stimulate the user to increase his or her gestural skills.

3 Game Application

GeStar Wars is a space shooter game that we developed in order to assess the benefit of feedback for in-game gestures. A screenshot of the game can be seen in Fig. 1. The game was implemented using the Ogre3D[1] graphics rendering engine under Ubuntu Linux[2]. In the game, the player controls a space-ship in the 2D sky. In order to navigate the ship s/he can use the arrow buttons of the Wiimote. The player can see the obtained scores on the bottom right of the screen. To perform a normal shot the player can use the B button of the Wiimote. Furthermore, we implemented three special features that can be activated only via gestures: a shield, which protects the space-ship for 8 seconds, and two more powerful special weapons (spread shot and flame shot).

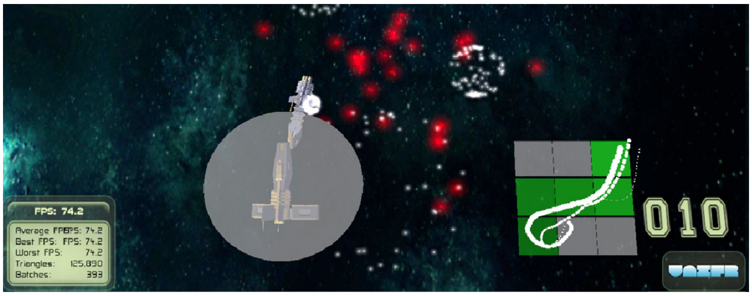

Fig. 1. Game with gesture feedback

The player has unlimited normal or special shots and can therefore perform the gestures as often as s/he wishes. The game allows for level customization in terms of duration and enemy space-ships. We implemented for our evaluation two different levels. The first level is a tutorial level, which is explained below (section 3.1), and the second one is a game level. For the game level we use 3 different types of enemies, each with different levels of difficulty. The game counts the points of the player. Indeed, the player gets points by destroying enemies and loses points when an enemy is shooting on the player's spaceship.

3.1 Tutorial Level

The tutorial level is a special level with no enemies which has the goal to help the player learning the gestures. At the beginning of the level the first gesture is illustrated (see **Error! Reference source not found.**) and the player can now test the first gesture. During this phase, the two other gestures are disabled. Afterwards, the

[1] `http://www.ogre3d.org`
[2] `http://www.ubuntu.com`

player can train and test the two other gestures. In the tutorial level, the player has 30 seconds to train each gesture several times.

4 Gestures

This section describes the gestures designed during the research project and those used during the evaluation. The gestures are defined through a path in the 3x3 grid and through its direction (starting and ending cells). The path is obtained by the instant vector of the filtered Wiimote x-z accelerometers. The sampling rate is 100 Hz. Gestures are segmented by the user with the A button. However, the maximum length of a gesture is 2 seconds and the first part of the gesture is truncated if it exceeds this length. The difficulty of a gesture is affected by its length, the amount of direction changes and the shape (a circle like Fig. 2. Table 1. Fig. 3g is easier to perform than a spiral like Fig. 2. Table 1. Fig. 3b). During the project we created 10 basic gestures (see Fig. 2. Table 1. Fig. 3). Since the path orientation is preserved, new gestures can be defined by simply rotating or mirroring the path of another gesture.

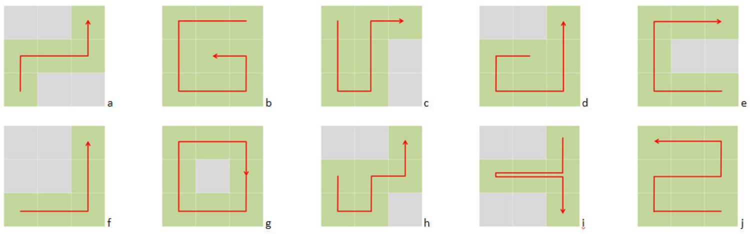

Fig. 2. Table 1. Fig. 3. Designed gestures

The gestures can be configured in GeStar Wars for special actions. As described above, in the game there are three special actions: the protection shield and two special shots (spread shot and flame shot).

For the evaluation we have chosen three of the above gestures; these three gestures are depicted with their respective actions in **Error! Reference source not found.**. We have chosen rather simple gestures in order to facilitate the learning of the gestures during the tutorial level in GeStar Wars.

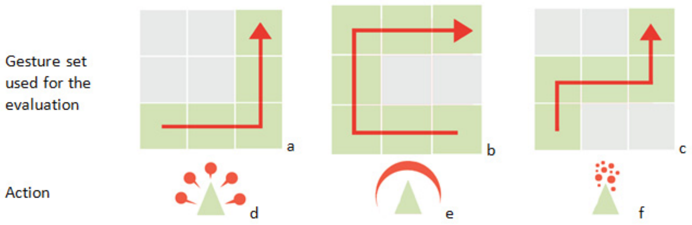

Fig. 4. The used gesture set for the evaluation *a* to *c*; the 3 gestures are illustrated from *d* to *f*

The first gesture is a horizontally mirrored "L". It permits to use the special spread shot. The second gesture is a "C" from the bottom to the top and permits to activate the shield. The third gesture is a 90° clockwise rotated "Z" and permits to activate the special flame shot.

4.1 Gesture Feedback

To support the user gesture learning, a particular multi-level gesture feedback strategy has been implemented. When the user performs a gesture which was not recognized or when the user press the A button on the Wiimote and suddenly releases it, all the possible gestures are illustrated in the top right corner. The illustrations of the gestures depict the directed path through the grid with a red arrow and highlight in green which cells of the grid has to be covered. Below the gesture illustration there is an illustration of the action that is associated to the gesture. **Error! Reference source not found.** shows the illustrations for the *a-d*, *b-e* and *c-f* gesture-action pairs, i.e., the three special actions of GeStar Wars. If a gesture was recognized with only one error, only the gesture illustration of that gesture is shown in the top right corner. In the tutorial level the gesture illustration that is currently learned is shown with a bigger size on the left bottom side.

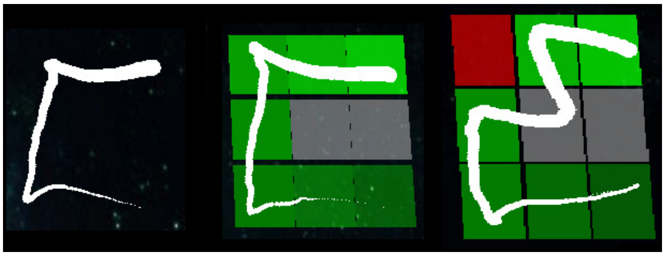

Fig. 5. Bottom right corner feedback: on the left side the gesture before its completion (A button release), in the middle the correct gesture and on the right side a gesture with an error

While the user performs a gesture, i.e., while the A button of the Wiimote is pressed, a white snake-like feedback is shown (see Fig. 4 left). This snake-like feedback represents the smoothed accelerometer values and depicts the path and direction of the ongoing gesture. Indeed, the snake-like feedback is smaller at the start and thicker at the end. Furthermore, an auto-scaled 3x3 grid is mapped under the snake-like feedback and it shows which cells were covered and in which order. Similar to the snake shape, the order of the covered cells is shown with different levels of green: the last covered cell is shown in a bright green and the first covered cell in a dark green. The cells that were not passed by the snake-like feedback are shown in gray (see Fig. 4).

In order to help users learning the gestures, our system is able to detect if there was only one cell missing by the snake-like feedback. In this particular case, the missed cell is highlighted in red in the bottom right corner feedback, in order to provide a suggestion to the user on how to correct his or her movement (see right side in Fig. 4).

To be able to give understandable feedback, the cells should be visited only once, although the gesture recognizer would allow using a cell several times in a gesture.

Furthermore, when a gesture is recognized, the Wiimote starts vibrating to give an additional non visual feedback.

5 Gesture Recognizer

The gesture recognizer was implemented in order to obtain a one-to-one mapping with feedback. This helps to give more accurate instant feedback and allows the user to understand how the system interprets his or her gestures. This gesture recognizer does not need machine learning; thus, it does not need system training and it can be quickly implemented. Due to its simplicity, this gesture recognizer is quite useful for prototypes.

The gesture recognizer exploits the Wiimote x-z accelerometer values. To facilitate the gesture segmentation, the user has to press the A button of the Wiimote while performing the gestures. In consequence, the gestures are stroke-like gestures.

While the user presses the button, the filtered values of the Wiimote are saved every 10 ms into a circular buffer. There is a predefined maximum gesture length. If this gesture length gets achieved, the system starts overwriting the oldest values. After the gesture performance, the system maps the auto-scaled 3x3 grid under the performed gesture. Because the Wiimote does not transmit a path but a series of points, a linear interpolation is performed in order to carefully check which rectangles are covered by the path. The grid auto-scaling allows the users to perform gestures as big or as small as they wish. To recognize a gesture, the algorithm compares the covered cells to the gestures in our gesture vocabulary. To accept a gesture, also the order on which the cells were covered must match. The algorithm allows having at the same time a gesture that can be contained in another gesture, for instance, a circle gesture and a double circle gesture.

With this system we are able to show to the user which cell, i.e., which part of a gesture, is not correct. This technique is helpful for learning the gestures and enables the possibility to show the error to the user and to mark with red the wrong cell in the grid (see Fig. 4). However, the designer should avoid having at the same time gestures with paths that differ by just one cell. Moreover, the current implementation is limited to two-dimensional gestures in order to map them in the 3x3 grid. An extension to three-dimensional gesture is possible but their representation in a three-dimensional cube would not be intuitive for the user.

6 Evaluation and Results

The purpose of these evaluations is to analyze if GeStar Wars gesture feedback influences the gesture learning and gesture performance of the user. We tested GeStar Wars with 12 users aged 22 – 54, 3 females and 9 males. We conducted a within subject user evaluation where all the users tested both conditions, without feedback and with feedback. In order to prevent the learning effect and the fatigue, half of the users started with feedback (group A) and half of the users started without feedback (group B).

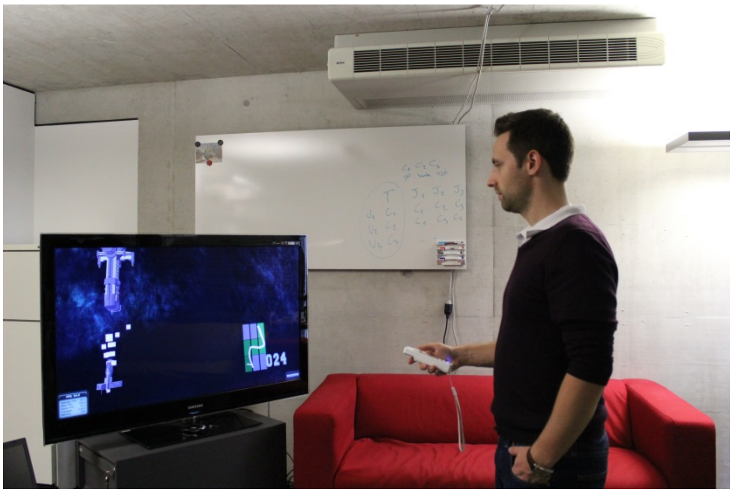

Fig. 6. Evaluation setup: user playing game level

For the user evaluation we used both the tutorial level and the game level. The evaluation started with the tutorial level, in particular with the spread gesture, followed by the shield and finally by the flame gesture. For each gesture the user has 30 seconds to practice. Before the tutorial, we gave to the participants some explications and we showed a video-tutorial that illustrates the tutorial session. The users did not have any experience with the game or the gestures before starting the tutorial. After the two tutorial sessions, we asked the volunteers to play twice a game level. We counterbalanced also here feedback presence (with and without). After each tutorial, we gave a slightly modified version of the questionnaire in the appendix C of the ISO 9241 part 9 for non-keyboard input devices. Since we were interested in how the user learned we did not analyzed the gesture accuracy during game level sessions. In fact, the game level requires more coordination to move the ship and perform the gestures at the same time, therefore, high variation could occur between skilled users and less practiced players. Nevertheless, the in-game evaluation has been useful as observation study.

We conducted a t-test to compare the recognized gestures for all the users (group A+B), with and without feedback. There was a significant positive effect of feedback on the percentage of recognized gestures: t(11) = 4.15, p = .0016. Significant difference was also found for the not recognized gestures (without the gestures containing one error): t(11) = -5.58, p = .00017. No significant difference was found between the gestures containing one error.

By comparing the total gestures and gesture attempts for each condition, we found that there was a significant effect of feedback on the gesture attempts: t(11) = -4.38, p = .0011. The average gesture attempts while using feedback in the tutorial level of 30 seconds was 25.4 attempts and while using no feedback it was 32 attempts. Thus, when no feedback was provided, users performed more gestures than with feedback; meanwhile, fewer gestures were recognized.

Table 2. The results of the user evaluation. Group A: user 1 to 6. Group B: user 7 to 12.

	With feedback			Without feedback		
User	**Recognized**	**With 1 error**	**Not recognized**	**Recognized**	**With 1 error**	**Not recognized**
1	16.0%	4.0%	80.0%	6.9%	0.0%	93.1%
2	16.7%	20.0%	63.3%	4.9%	7.3%	87.8%
3	20.0%	0.0%	80.0%	0.0%	3.2%	96.8%
4	35.3%	23.5%	41.2%	30.4%	26.1%	43.5%
5	22.2%	11.1%	66.7%	9.1%	6.1%	84.8%
6	30.0%	20.0%	50.0%	26.1%	21.7%	52.2%
7	50.0%	7.1%	42.9%	23.1%	12.8%	64.1%
8	33.3%	22.2%	44.4%	36.0%	8.0%	56.0%
9	10.5%	10.5%	78.9%	2.6%	2.6%	94.7%
10	48.0%	8.0%	44.0%	2.8%	8.3%	88.9%
11	37.5%	12.5%	50.0%	15.8%	10.5%	73.7%
12	36.1%	22.2%	41.7%	13.0%	19.6%	67.4%
Avg	29.6%	13.4%	56.9%	14.2%	10.5%	75.2%
SD	12.7pp	8.0pp	15.9pp	12.0pp	8.1pp	18.3pp

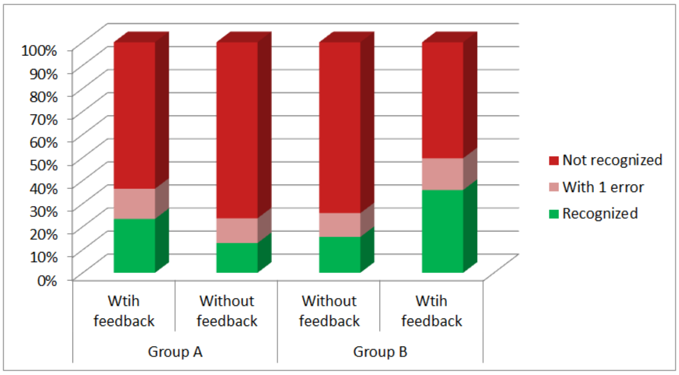

Fig. 7. Diagram showing gesture recognition rates

For both groups we found significant difference between the two conditions: with and without feedback. Comparing the two groups, we discovered that the group A was performing worse when using no feedback, even if they were already trained (first and second bar in **Error! Reference source not found.**). We suppose that this is due to the short learning time; thus, the users do not know the gestures well enough after one tutorial in order to perform them without feedback. The group B was then performing a lot better when they used feedback (third and fourth bar in **Error! Reference source not found.**). Therefore, in both cases feedback was increasing the user accuracy while learning the gestures.

In a questionnaire given at the end of the experience, all of the twelve users preferred to have feedback for the tutorial level. However, for the game level only 8 users preferred to have feedback. The 4 users that preferred no feedback during the game level explained that they had no time to look at feedback while playing the game.

Concerning the ISO 9241 part 9 questionnaire at the end of each tutorial, we found no statistical significance in terms of cognitive load. The only significant difference (Wilcoxon signed-rank test) was found for the arm fatigue. The tutorial with no feedback was found more tiring. This can be due to the fact that the user tried to make more gestures while having no feedback.

7 Conclusion and Future Work

GeStar Wars, a space shooter game with stroke-like gestures performed with a Wiimote, was presented in this article. The novelty of this game is the proposed feedback, which is directly linked to the gesture recognizer. Feedback is composed of a snake-like stroke, which represents the smoothed accelerometer values of the gesture, and an auto-scaled 3x3 grid, on which the gesture is mapped. A gesture is therefore defined by the cells of the grid and their covering order. The gesture recognizer uses the accelerometer values and therefore is not based on direct pointing.

By analyzing the results of the evaluation, we found significant difference on the percentage of recognized gestures when using or not feedback. In general the users performed better when using feedback. We were also able to show that the performance is increasing when feedback is added and decreasing when feedback is removed. We found also that the users performed significantly more gesture attempts without feedback. Besides the results, all the users preferred to have feedback for the tutorial level. The user has to learn how the system interprets the gestures and, for this purpose, feedback is helpful. In our evaluation we noticed that the training session (the two tutorials) is not long enough to learn the gestures. Even if the users were quite better at the end of the training and in the game level, a more trained person, as the authors, for instance, have still a much higher acceptance rate of the gestures. We suggest therefore training a gesture much longer. Some of the users had problems performing the gestures when several enemies came and they were under "stress". Once the gestures are learned, feedback may distract the user from the enemies in the game. We remarked this also in the evaluation, since, after the two tutorial sessions, we let the user play the game level with the enemies. Some of the users looked more at the gesture feedback than at the rest of the game. Nevertheless, the implemented feedback works well for the tutorial level where the user has more time to look at feedback. In fact, if the user has to look at the enemies s/he has not much time to look at feedback in detail.

For future work it would be interesting to create a mixed mode in order to support the user with feedback when s/he did the gestures wrong for two consecutive times and remove feedback when s/he starts doing the gestures correctly. Moreover, it would be interesting to create other visual feedbacks to help the users learning the gestures. We could imagine to illustrate an animation of the Wiimote showing the gesture, in the case where the user did an error. It would also be interesting to extend feedback by other modalities, such as sound and/or improving the vibration feedback of the Wiimote.

References

1. Subrahmanyam, K., Greenfield, P.M.: Effect of video game practice on spatial skills in girls and boys. J. Appl. Dev. Psychol. 15, 13–32 (1994)
2. Kela, J., Korpipää, P., Mäntyjärvi, J., Kallio, S., Savino, G., Jozzo, L., Marca, S.: Di: Accelerometer-based gesture control for a design environment. Pers. Ubiquitous Comput. 10, 285–299 (2005)
3. Kratz, S., Rohs, M.: A $3 gesture recognizer. In: Proceedings of the 15th International Conference on Intelligent User Interfaces, IUI 2010, p. 341. ACM Press, New York (2010)
4. Liu, J., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uWave: Accelerometer-based personalized gesture recognition and its applications. Pervasive Mob. Comput. 5, 657–675 (2009)
5. Rehm, M., Bee, N., André, E.: Wave Like an Egyptian — Accelerometer Based Gesture Recognition for Culture Specific Interactions
6. Bau, O., Mackay, W.E.: OctoPocus: A Dynamic Guide for Learning Gesture-Based Command Sets. In: Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology, UIST 2008, p. 37. ACM Press, New York (2008)
7. Bragdon, A., Uguray, A., Wigdor, D., Anagnostopoulos, S., Zeleznik, R., Feman, R.: Gesture play. In: ACM International Conference on Interactive Tabletops and Surfaces, ITS 2010, p. 39. ACM Press, New York (2010)
8. Bragdon, A., Zeleznik, R., Williamson, B., Miller, T., LaViola, J.J.: GestureBar. In: Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, p. 2269. ACM Press, New York (2009)
9. Li, P., Bouillon, M., Anquetil, E., Richard, G.: User and System Cross-Learning of Gesture Commands on Pen-Based Devices. In: Kotzé, P., Marsden, G., Lindgaard, G., Wesson, J., Winckler, M. (eds.) INTERACT 2013, Part II. LNCS, vol. 8118, pp. 337–355. Springer, Heidelberg (2013)
10. Freeman, D., Benko, H., Morris, M.R., Wigdor, D.: ShadowGuides: Visualizations for In-Situ Learning of Multi-Touch and Whole-Hand Gestures. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, ITS 2009, p. 165. ACM Press, New York (2009)
11. Schlömer, T., Poppinga, B., Henze, N., Boll, S.: Gesture recognition with a Wii controller. In: Proceedings of the 2nd International Conference on Tangible and Embedded Interaction, TEI 2008, p. 11. ACM Press, New York (2008)
12. Wu, J., Pan, G., Zhang, D., Qi, G., Li, S.: Gesture Recognition with a 3-D Accelerometer, pp. 25–38 (2009)