A Grammar-Based Framework for Rehabilitation Exergames

Victor Fernandez-Cervantes(✉), Eleni Stroulia, and Benjamin Hunter

University of Alberta, Edmonton, AB, Canada
{vf,stroulia,bkhunter}@ualberta.ca

Abstract. Numerous serious exergames advocate the use of engaging avatars to motivate a consistent exercise regimen. However, the process of specifying the prescribed exercise, implementing it as avatar animation, and developing an accurate feedback-providing mechanism is complex and requires a high level of expertise in game engines, control languages, and hardware devices. Furthermore, in the context of rehabilitation exergames, the requirements for accurate assessment and timely and precise feedback can be quite stringent. At the same time, the Kinect^{TM} motion-capture sensor offers a natural interface to game consoles, and its affordability and wide availability represents a huge opportunity for at-home exergames. In this paper, we describe our work towards a system that envisions to simplify the process of developing rehabilitation exergames with Kinect^{TM}. The system relies on a language for specifying postures and movements between them, and includes an editor that enables rehabilitation therapists to specify the prescribed exercise, by editing a demonstration of the exercise. This exercise-specification grammar is used to drive the animation of an avatar and the provision of quality feedback, by comparing the player's postures (as captured by the Kinect^{TM}) against those of the coaching avatar and the grammar.

Keywords: Kinect-based gameplay · Interface · Serious games · Rehabilitation

1 Introduction

The concept of "serious games" refers to digital games whose purpose is more than entertainment [1]. The core intuition behind the serious-game paradigm is that, when learning tasks are embedded within a gameplay scenario, learners, motivated by the mechanics of gameplay such as "scoring points", "clearing levels", and "getting badges", spend more time learning, which facilitates the acquisition of new knowledge and skills.

A particularly interesting type of serious games are "exergames", i.e., games designed to encourage physical activity towards improving physical ability and

© IFIP International Federation for Information Processing 2016
Published by Springer International Publishing AG 2016. All Rights Reserved
G. Wallner et al. (Eds.): ICEC 2016, LNCS 9926, pp. 38–50, 2016.
DOI: 10.1007/978-3-319-46100-7_4

fitness, or towards systematizing athletes' training, or towards rehabilitation from injury or other health challenges. The development of exergames is a quite demanding software-engineering task: in addition to requiring the design of engaging avatars, animations and game mechanics, it also demands knowledge about specialized hardware sensors and controllers, game engines, algorithms for processing the sensor signals, producing feedback for the user and controlling the game state. Even more importantly, it relies on the specialized domain knowledge of exercise experts, such as trainers and physical therapists.

Exergames have received substantial attention recently, as controllers that use full-body motion, such as the very popular and relatively inexpensive Wii and Kinect^{TM}, have become increasingly available at-home. The latter, in particular, represents an extremely attractive platform for exergames [9] since it enables adequate skeleton tracking, and its SDK and user community offer substantial software-development support. Recognizing this opportunity, a variety of exergames have been developed that cover a broad design spectrum: on one hand, one can find complex systems with no evident gameplay [6]; on the other hand, some games [7,8] offer engaging gameplay but with restricted and limited movements, and, therefore, limited potential for physical conditioning.

It is our belief that the Kinect^{TM}-based gameplay for exergames suffers from a feasibility gap: today, there is no toolkit to support the integration of personalized animation, guidance feedback, and valid exercise assessment. As a result, even though the sensor is cost-effective, widely available, and sufficiently accurate, its potential is not yet fully met. Motivated by this position, the key objective of this work is to introduce Avatar Grammar Animation System (AGAS), our Kinect^{TM}-based toolkit for supporting the development of rehabilitation exergames in Unity. Our system offers an editing tool that enables casual computer users, with expertise in physical exercise but no software-engineering expertise, to specify an exercise script, including key postures and the movement between them. This editor, in effect, enables users to annotate a pre-recorded demonstration of the exercise with rules about the angles of the important joints in each step of the exercise. The resulting exercise script, represented in terms of the underlying AGAS grammar, an extension of the grammar reported in [8], is then used by the Unity game engine to (a) animate a coach avatar that demonstrates the exercise at run-time during gameplay, and (b) provide timely and accurate feedback to the player about his/her posture and movement.

The rest of this paper is organized as follows. Section 2 places our work in the context of related research, reviews Kinect^{TM} as the underlying hardware of our system, and motivates the use of fuzzy logic for assessing the user's movements. In Sect. 3, we describe the architecture of our system and our experience with it to date. Finally, in Sect. 4, we close with a summary of the key contributions of our system and our plans for future work.

2 Related Research and Background

Exergames: In 2006, the Wii console evolved the gameplay paradigm with an interactive control-system that established a rapport between the player and a

digital avatar [2]. This rapport was shown to motivate exercise consistency and perseverance of seniors, who reported increasing levels of enjoyment during the 6-weeks training program with Wii Fit Plus games [3]. Another study reported similar results with overweight children who improved their exercise habits playing a virtual running game, featuring a slim and toned avatar design [4].

In late 2010, the affordable motion-capture Kinect^{TM} changed the gameplay once again to a natural body-motion interface. Even though the videogame console, Xbox, did not exploit this new capability –gameplay in games such as JUST DANCE[1] or ZUMBA fitness[2] is the same across all the main videogame consoles, and does not take advantage of the unique features of each console– the Kinect^{TM}'s potential for serious games has been amply demonstrated. In [5], the data from multiple tracking devices is combined to recognize dance patterns, using a Hidden Conditional Random Fields (HCRF) classifier. In [6], "seated Tai Chi" is presented as a Kinect-based physical rehabilitation exercise. The system was designed for patients with movement disorders and assessed their ability with a very simple measure: it evaluated whether some angles of interest in the patient skeleton were equal (within a threshold) to some pre-defined values. As a result the system can only give very simple feedback on eighteen postures. The same control system with more innovative game-flow design was presented in the sorcerer's apprentice [7], which was designed for patients with SIS (Subacromial Impingement Syndrome). The prescribed movements are specified as boundaries preconfigured by the therapist. The gameplay allows the patient to focus on playing, while achieving the goals in the rehabilitation exercises. In our own previous work [8], we focused on a more realistic mechanism for providing high-quality feedback in rehabilitation serious games. To that end, we developed a fuzzy grammar for evaluating the correctness of postures and movements. Our original grammar was, however, limited to analyzing mostly static postures and dynamic transitions through changes of a single angle between three joints. We also developed an intutitive user-interface feedback guide: a fill-up bar, attached to the corresponding joint, which fills up following the movement of the player and presents the correct angle with an arrow. In this paper, we describe an extension of our original grammar that can capture more complex exercises (such as TaiChi) and we develop an editor to enable exercise experts to define exercise scripts in this grammar.

Avatar Animation: The topic of avatar animation is quite vast and, in this section, we cannot but mention only an eclectic collection of works, on two broad topics. On the topic of movement animation, there is work on inverse kinematics for maintaining balance [10]; skeleton animation with walking cycles [11]; falling and landing with concern for self-preservation [12]; and realistic mass and balance properties of physical characters [13]. On the topic of the character appearance, there is work on animating non-human characters based on skeleton semantics [14]; representing avatars as arbitrary 3D mapping points [15] or with visually highly realistic bodies [16] and facial expressions [17]; and animating

[1] http://just-dance.ubi.com.
[2] http://zumbafitnessgame.com/.

3D puppets based on real-time motion capture [18]. In the context of serious exergames for rehabilitation, our interest in avatar animation stems from the need to communicate in an intuitive and unambiguous manner the movements to be mimicked by the player. In effect, our goal is to develop a system that will enable non-computer experts to animate a precise coach avatar demonstrating the prescribed exercise and providing timely and accurate feedback to the player.

A few platforms have been proposed to facilitate this task. The Flexible Action and Articulated Skeleton Toolkit (FAAST) [19] is a middleware that integrates full body control. Unfortunately, the gestures captured with the Kinect^{TM} are only substitutes for keyboard commands or a virtual mouse. XDKinect [20] is a similar toolkit, with an online framework service to recognize gestures. In [21], a multimedia environment for children rehabilitation is described. The system controls an avatar in Second Life to demonstrate the therapy exercises. The system is limited in terms of the exercises it can cover: the exercise postures focus on a single joint angle and so does the assessment. The motion-tracking evaluation system presented in [22] uses Unreal Engine 4 to compare the player's skeleton as recorded with the Kinect^{TM} sensor with the in-game avatar. This system requires a developer to program the avatar and offers very coarse-grained feedback regarding the correctness of the player's movements: simply red and green colors over the avatar's joints. This simple feedback makes it almost impossible to discern the specific corrections required in each posture. The Dual-Task Tai Chi game suffers from similar limitations, but with more interesting game play [23], using Kinect^{TM}-based full-body control with a cognitive task, i.e., 4×4 Sudoku. In this game, the user has to reach a specific posture in order to select a number with the right hand or foot, and then place this number on the Sudoku grid with the left hand or foot. The posture sequence during the game are similar to tai-chi.

The Kinect^{TM} Sensor: The Kinect^{TM} V2, on which AGAS is based, presents many improvements over its predecessor [24]. It is composed by infrared and color camera, with a resolution of 512×424, and 1920×1080 respectively. The sensing process is orchestrated by a fast clock signal whose strobes an array of three laser diodes, which simultaneously shine through diffusers, bathing the scene with short pulses of infrared light. The sensor also calculates the ambient-lighting, the final image invariant to lighting-changes. The accuracy of the Kinect^{TM} V2 in computing the joints' positions is lower than motion-capture systems, however it is "good enough" for some of the regimen exercises, or therapy. the discrepancy could go from 13 mm to 64 mm [25].

The Kinect^{TM} SDK version 2.0 estimates the position and orientation of 25 joints, organized in a hierarchy, centered at the spine base (SB), as shown in Fig. 1(a). This hierarchy causes difficulties with several postures, due to occlusion or, more generally, lack of information regarding the root of a particular joint and its hierarchical rotation, i.e., the amount of rotation in the 3D space that the joint inherits from its parent joint.

Fuzzy Logic: In order to assess the player's exercise style and provide feedback to motivate and improve it, an efficient mechanism for continuously comparing

the player's actual posture against the "correct" one is required. In this work, similar to our previous work [8], we adopt a fuzzy-logic paradigm for that purpose, in order to avoid simplistic angle comparisons (as used in [6]). The fuzzy-logic paradigm considers that uncertainty is unavoidable when concepts are imprecisely expressed in natural language: in the context of the rehabilitation-exercise specification, the concepts of "correct" and "incorrect" posture are expressed with a real value between "1" and "0" indicating a degree of correctness.

3 Software Architecture

The AGAS system consists of three components: (a) a recording component through which an *expert demonstration* of the exercise is captured in a form that can be viewed and manipulated in Unity;[3] (b) an editor, implemented in Unity, through which a exercise expert reviews the demonstration to produce an *exercise script*, including *key postures*, their *important joints*, and the transitions between them; and (c) a game-playing engine, also implemented in Unity, during which the player's movement is compared against the coach avatar, i.e., the exercise script animation, and the rules around the important key-posture joints.

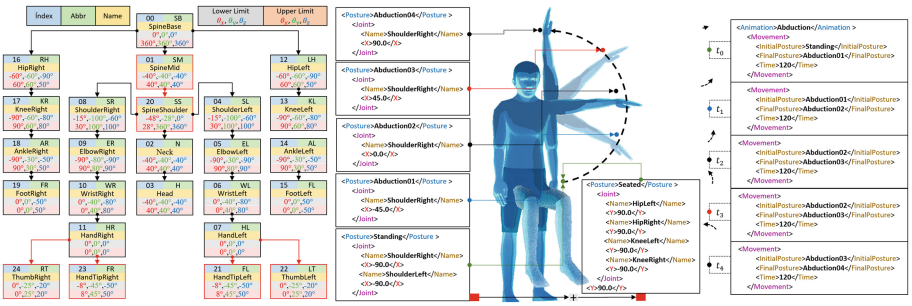


Fig. 1. (a) Kinect^{TM} skeleton tracking SDK 2.0 (Position, Orientation, Hierarchy, and Rotation); (b) Exercise-script animation

3.1 Exercise Demonstration and Recording

The motion-capture-and-recording component records 30 fps, each of which includes the complete information regarding all joints shown in Fig. 1(a). Each frame record includes (a) the frame timestamp in milliseconds, with the first

[3] The AGAS recorder is based on the examples of https://www.assetstore.unity3d.com/en/#!/content/18708 for how to use Kinect^{TM} V2 in Unity, including examples for how to start the camera, how to record data, and how to control the avatar based on the recording.

frame defining time 0; and (b) a sequence of joint records, each one consisting of the joint's identification number, and its orientation in three-dimensional space, centered at the 0_{th} joint, at the *SpineBase*.

3.2 The Exercise-Script Editor

In our AGAS system, an exercise specialist can review and edit the *exercise demonstration*, through the editor, shown in Fig. 3. The AGAS editor reads as input the demonstration frame sequence and provides a user interface through which an exercise expert, can review the the demonstration and identify the exercise *key postures*, selecting the appropriate frames where these postures are demonstrated.

Reviewing Avatar Postures: Through the AGAS editor, the exercise expert can replay, stop, move to next and previous frames of the demonstration record. When reviewing the avatar posture in a particular frame, the user can inspect the posture from multiple camera views and can review each avatar joint.

As shown in Fig. 1(a), the avatar is a rigid body with 25 interconnected joints. The orientation of each joint is relative to its parent joint in the skeleton hierarchy of Fig. 1(a). This implies that a change in the orientation of a joint affects the positions of all its descendant joints. For each joint j_i, in each frame of the recorded exercise demonstration, the following information is available:

$$j_i = \{Name, \theta_x^i, \theta_y^i, \theta_z^i\} \tag{1}$$

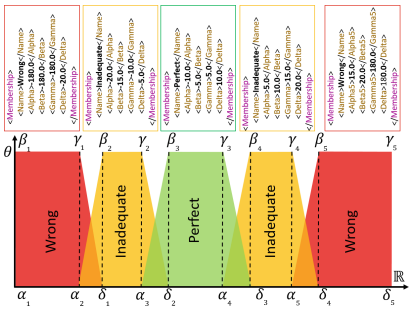
where *name* is the joint's label (and implicitly its position in the joint hierarchy) in the Kinect^{TM} skeleton, and the angles $\theta_x, \theta_y, \theta_z$ correspond to the orientation of the joint in the specified axes, in the frame under examination.

Specifying Key Postures and Transitions: The main purpose of the editor component is to produce an *exercise script*, based on the expert's demonstration of the exercise. The exercise script consists of a succession of *key postures*, starting with the *initial posture* demonstrated by the expert. The intuition behind the definition of the exercise script is that the player going through the exercise **has to** reach some key postures, which are defined in terms of their *important posture joints*, i.e., a subset of joints that have to be at precise orientation angles in this key posture. These major posture joints move from one key posture to the next, according to the demonstrated timing. The exercise expert, using the editor, (a) identifies the key postures; (b) specifies the major joints for each posture; (c) reviews and corrects the precise orientation angles of the major joints - in case the demonstration is not perfect; and, implicitly, (d) records the timing between these major postures.

The various elements involved in the specification of an exercise script are illustrated in Fig. 1(b). *Posture* elements specify the static skeleton configurations that the user must reach during the exercise. *Movement* elements specify the timing of the dynamic transition between two consecutive postures, expressed in the number of lapsed frames between the two postures. For every exercise

script, there are two special postures: the *initial posture* is the first skeleton configuration of the recorded demonstration, and the *final posture* is the final skeleton configuration, correspondingly. The two may be the same, in fact they often are for exercises that start and finish at the same position, after going through a sequence of other intermediate postures. Each posture is characterized by a descriptive *name* element, and its unique *id* attribute. Each posture element is composed of a set of *joint* elements, a proper subset of the 25 Kinect^{TM}-skeleton elements, whose exact orientations are important for the posture to be considered correctly achieved. For each joint, its name is specified (as shown in Fig. 1(a)) as well as some (or all) of the *X-axis, Y-axis, Z-axis* orientation angles, relative to its parent joint.

```
<posture initial=true id=1>
    <name>posture name</name>
    <joint important=true>
        <name>joint name</name>
        <axis>angle</axis>
    </joint>
    <joint> ... </joint>
    ...
</posture>
```




```
<movement>
    <source><ref:posture ref=1></source>
    <target><ref:posture ref=100></target>
    <time>frames</time>
</movement>
```

Fig. 2. The exercise script language (Color figure online)

The key postures are identified by the exercise expert as he/she reviews the exercise demonstration (see Sect. 3.1). As the expert moves through the frames of the demonstration record, he/she recognizes a key posture and, through the user interface in Fig. 3, selects the important joints, i.e., the defining joints for the posture. If a particular joint is not in a "perfect" position, the expert may also edit its orientation angles. Through this interaction, the key postures of the exercise are collected in the exercise script. The movement elements are inferred implicitly, based on the data between two consecutive key postures. The initial and final key postures are identified, by default, as the postures on the initial and final frames of the demonstration record. Finally, it is important to note that the specification of an *important joint* in a *key posture* gives rise to a corresponding fuzzy *assessment rule*, to be evaluated during game play at run time.

Coach-Avatar Animation: The coach avatar mimics the demonstrated exercise as a prearranged animated sequence, described in the exercise script.

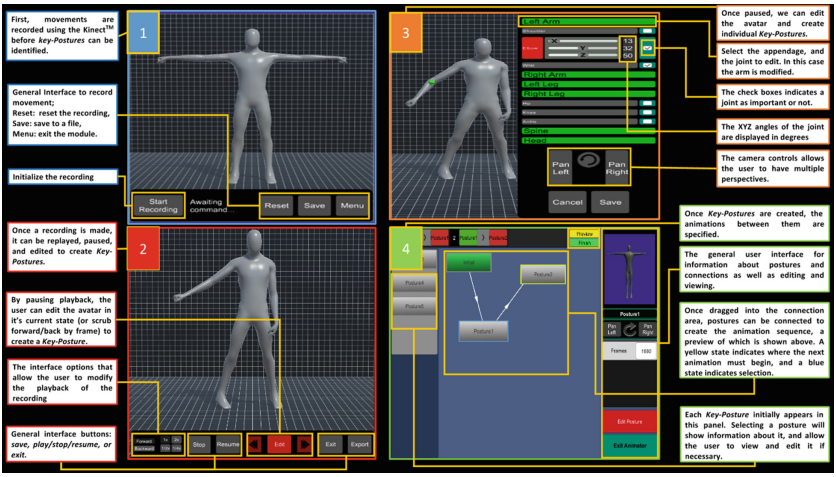


Fig. 3. The AGAS exercise-script editor

The coach avatar is composed of 33 joints. In addition to the twenty-five main joints of the Kinect^{TM} avatar, eight more joints are used for the animation of muscle movement, in key areas such as hips, legs, and shoulders, enabling a more realistic appearance of body movement.

The coach avatar initially appears with the standard T-Posture; in this posture, all Kinect^{TM}-skeleton joints are assumed to be in their corresponding 0,0,0 position. The animation process is controlled by the exercise script, as follows.

(a) At each step, the coach avatar moves from one key posture (source) to the next (target).
(b) The transition from the source to the target key posture is completed in the number of frames prescribed by the corresponding animation element in the exercise script.
(c) All major joints of the source key posture move synchronously until they reach their orientation in the target key posture. In principle, there are two directions of movement between any two angles; the direction of the movement is chosen so that the distance the joint travels is minimized. In order to produce a naturalistic animation, corresponding joints in subsequent key postures should not be more than 90° away (in any angle).
(d) Finally, the deltas of the joints' movements, between two frames, in all axes are equal. In effect, the animation component extrapolates a number of intermediate postures, at a rate of 30 fps, between every two consecutive key postures, to create the appearance of a smooth and fluid movement.

The animation process starts with the default T-posture which transitions to the first key posture of the exercise, and then iterates through all key postures until the last one.

An interesting feature of the AGAS component for exercise-script animation is that the same exercise script can potentially be applied to different original postures. For example, one could apply a script for upper-arm movement to an original sitting posture, for wheelchair-bound users. As long as the major joints of the key postures in the exercise script do not involve movements of the leg joints, the exercise could be demonstrated by a standing or a sitting coach avatar.

The AGAS editor user interface, shown in Fig. 3, enables all the design-time functionalities, i.e., exercise recording and replaying, identifying and reviewing key-postures, and specifying the transitions between them. In Fig. 3, subsection 1 shows the recording interface, where the movements are recorded and saved. The playback interface is shown in subsection 2, where the user has the ability to pause the playback, adjust its speed and direction, scrub frame-by frame, and export all key postures created. When playback is paused the user is also given the option to edit the avatar in its current position and save a key posture. This interface is shown in subsection 3, where the user may signify joints as important as well as select them to observe and modify the corresponding X, Y, and Z angles. Camera controls allow multiple views of the posture and a save option lets the user save the current pose as a key posture. Finally, subsection 4 shows the interface for reviewing and animating created key postures. Here the user can review and edit each posture, or finally preview and specify the transitions between two key-postures to create the animation script for gameplay.

3.3 Kinect-Based Gameplay

During gameplay, the coach avatar demonstrates the exercise in the context of the game background, the user performs the exercise and his/her avatar reflects his/her movements right next to the coach avatar, and the assessment component evaluates the correctness of the player's performance and provides appropriate feedback, through highlighting of the user's skeleton's joints (see Fig. 1(b)). We have explained the coach avatar demonstration in Sect. 3.2 and the replay of the Kinect^{TM}-captured user movement in Sect. 3.1. In this section, we describe the assessment component of our AGAS system.

The AGAS exercise-assessment component performs two types of evaluation of the user's performance, according to two measures: (a) the frame-by-frame similarity of the user's skeleton to the coach-avatar skeleton, and (b) its adherence of the user's skeleton to the rules around the orientation of the major joints of each key posture.

Conformance to Key Postures: In general, the complete assessment of a *key posture* relies on a collection of rules, each one corresponding to a major joint of the posture, J_2, and its angle relative to its parent and child joints, J_1, and J_3. The joints J_1 and J_3 form a interior angle with J_2, and the rule defines the correct angles θ among these joints in the corresponding axes: $\theta_x(left - right)$, $\theta_y(up - down)$, $\theta_z(front - back)$. The assessment rules are fuzzy: they specify smooth transition between correct and incorrect joint orientation, as opposed

to defining a sharp boundary to separate these two states. For each important joint, a fuzzy rule, illustrated by the five overlapping trapezoid functions, shown in Fig. 2, defining a range of transitions from perfect to wrong. At run time, the game-playing component continuously evaluates the rules for all major joints for the next anticipated key posture, and identifies the frames at which each of the rules is met "perfectly" or "adequately" or "not at all" and provides feedback by annotating the joints with a color, ranging from green(perfect) to yellow(inadequate) and red(wrong), with a smooth color transition in between functions.

Frame-by-Frame Dynamic Angle Similarity: In parallel with the rule-based assessment of the key-posture joints, a second assessment measure is applied to the major arm and leg bones of the player. At each frame, the orientations of these bones in the player's avatar skeleton are compared against the orientations of the corresponding bones in the coach avatar skeleton.

During game play, the feedback component uses the frame-by frame assessment to highlight the arms and legs of the user's avatar. A configurable "difficulty" parameter modifies a threshold which defines what is perfect (green), neutral (yellow), or incorrect (red). For each compared bone, the average of the absolute differences of the three corresponding angles is compared against this threshold to determine the color shown to the user; the closer to 0 the absolute average difference is, the better the player has matched the coach.

3.4 Experience and Reflection

The AGAS tool enables the development of the basic elements of a serious exercise game, based on a demonstration of the game exercise. The display in Fig. 4 shows the product of this process, placed on a background. At this point, the game is playable, and it enables valid exercise by giving basic feedback on its correctness; yet, further work is needed to make it engaging and motivating. To complete the game, the aesthetic elements of the game have to be developed, including the background visuals, the music, and the body(ies) of the coach and player avatar. This remaining development is primarily the responsibility of designers; in effect, AGAS systematizes and simplifies the developers' tasks, so that an exercise expert can actually accomplish them based on their domain knowledge. To date, we have used AGAS to develop a set of simple upper-body exercises, similar to the ones found at http://eldergym.com/elbow-exercises.html. We have also developed a small TaiChi game, consisting of few simple TaiChi moves. The former game was relatively easy to develop, since the key postures are relatively clear to identify (in fact for most of the above exercises the eldergym web site identifies the key postures with still pictures), each posture consists of few important joints, and the orientation angles of these joints are simple to describe. In fact for most of these exercises the joints' orientations between key postures change only in one dimension. The TaiChi game is, however, much more challenging: the key postures are much more complex and multiple joints move between any two of them.

Fig. 4. Run-time posture assessment and feedback (Color figure online)

4 Conlcusions

In this paper we described AGAS, a tool for supporting the development of serious exergames, that can be used for rehabilitation purposes. These games must communicate in an intuitive and unambiguous manner the movements of the player's rehabilitation regimen, as prescribed by an exercise expert. In developing AGAS, our goal was to enable non-computer experts to animate a precise coach avatar demonstrating the prescribed exercise and providing timely and accurate feedback to the player. At the core of our tool is a simple grammar for postures and transition movements between them. Specifications of an exercise in this grammar can be constructed by an exercise expert using the AGAS editor to inspect and annotate a demonstration of the exercise. These specifications are then fed into the AGAS game-playing component, which is responsible for observing the player's movements, comparing them to the specification as well as to a coach-avatar simulation and providing feedback to the player so that he can improve his posture and movement.

In the future, we plan to improve the feedback component to provide more accurate feedback with respect to timing, and we will extend the editor to support the exercise experts with hints as to which the best key postures might be.

References

1. Susi, T., Johannesson, M., Backlund, P.: Serious games: an overview (2007)
2. Karoussos, K.: Mii & you. In: Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts, pp. 496–498. ACM (2008)
3. Nicholson, V.P., McKean, M., Lowe, J., Fawcett, C., Burkett, B.: Six weeks of unsupervised Nintendo Wii Fit gaming is effective at improving balance in independent older adults. J. Aging Phys. Act. **23**(1), 153–158 (2015)
4. Li, B.J., Lwin, M.O., Jung, Y.: Wii, myself, and size: the influence of Proteus effect and stereotype threat on overweight children's exercise motivation and behavior in exergames. Games for Health Res. Develop. Clinical Appl. **3**(1), 40–48 (2014)
5. Kitsikidis, A., Dimitropoulos, K., Douka, S., Grammalidis, N.: Dance analysis using multiple kinect sensors. In: 2014 International Conference on Computer Vision Theory and Applications (VISAPP), vol. 2, pp. 789–795. IEEE (2014)

6. Lin, T.Y., Hsieh, C.H., Lee, J.D.: A kinect-based system for physical rehabilitation: utilizing tai chi exercises to improve movement disorders in patients with balance ability. In: 2013 7th Asia Modelling Symposium (AMS), pp. 149–153. IEEE (2013)
7. Fikar, P., Schoenauer, C., Kaufmann, H.: The Sorcerer's Apprentice A serious game aiding rehabilitation in the context of Subacromial Impingement Syndrome. In: 2013 7th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth), pp. 327–330. IEEE (2013)
8. Fernandez-Cervantes, V., Stroulia, E., Oliva, L. E., Gonzalez, F., Castillo, C.: Serious games: rehabilitation fuzzy grammar for exercise and therapy compliance. In: 2015 IEEE Games Entertainment Media Conference (GEM), pp. 1–8. IEEE (2015)
9. Obdrzalek, S., Kurillo, G., Ofli, F., Bajcsy, R., Seto, E., Jimison, H., Pavel, M.: Accuracy and robustness of Kinect pose estimation in the context of coaching of elderly population. In: 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 1188–1193. IEEE (2012)
10. Baerlocher, P., Boulic, R.: An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. Vis. Comput. **20**(6), 402–417 (2004)
11. Multon, F., France, L., Cani-Gascuel, M.P., Debunne, G.: Computer animation of human walking: a survey. J. Vis. Comput. Anim. **10**(1), 39–54 (1999)
12. Dykes, S.B.: U.S. Patent No. 8,228,336. U.S. Patent and Trademark Office, Washington, DC (2012)
13. Kenwright, B.: Real-Time Physics-Based Fight Characters
14. Wang, X., Ma, Q., Wang, W.: Kinect driven 3D character animation using semantical skeleton. In: 2012 IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS), vol. 1, pp. 159–163. IEEE (2012)
15. Rhodin, H., Tompkin, J., In Kim, K., Varanasi, K., Seidel, H.P., Theobalt, C.: Interactive motion mapping for realtime character control. Comput. Graph. Forum **33**(2), 273–282 (2014)
16. Teran, J., Sifakis, E., Blemker, S.S., Ng-Thow-Hing, V., Lau, C., Fedkiw, R.: Creating and simulating skeletal muscle from the visible human data set. IEEE Trans. Vis. Comput. Graph. **11**(3), 317–328 (2005)
17. Kalra, P., Magnenat-Thalmann, N., Moccozet, L., Sannier, G., Aubel, A., Thalmann, D.: Real-time animation of realistic virtual humans. IEEE Comput. Graph. Appl. **18**(5), 42–56 (1998)
18. Leite, L., Orvalho, V.: Anim-actor: understanding interaction with digital puppetry using low-cost motion capture. In: Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology p. 65. ACM (2011)
19. Suma, E.A., Lange, B., Rizzo, A.S., Krum, D.M., Bolas, M.: Faast: the flexible action and articulated skeleton toolkit. In: 2011 IEEE Virtual Reality Conference (VR), pp. 247–248. IEEE (2011)
20. Nebeling, M., Teunissen, E., Husmann, M., Norrie, M.C.: XDKinect: development framework for cross-device interaction using kinect. In: Proceedings of the 2014 ACM SIGCHI Symposium on Engineering Interactive Computing Systems, pp. 65–74. ACM (2014)
21. Abdur Rahman, M., Qamar, A.M., Ahmed, M.A., Ataur Rahman, M., Basalamah, S.: Multimedia interactive therapy environment for children having physical disabilities. In: Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval, pp. 313–314. ACM (2013)
22. Alabbasi, H., Gradinaru, A., Moldoveanu, F., Moldoveanu, A.: Human motion tracking evaluation using Kinect V2 sensor. In: E-Health and Bioengineering Conference (EHB) 2015, pp. 1–4. IEEE (2015)

23. Kayama, H., Okamoto, K., Nishiguchi, S., Nagai, K., Yamada, M., Aoyama, T.: Concept software based on Kinect for assessing dual-task ability of elderly people. Games for Health Res. Develop. Clin. Appl. **1**(5), 348–352 (2012)
24. Butkiewicz, T.: Low-cost coastal mapping using Kinect v2 time-of-flight cameras. In: Oceans-St. John's 2014, pp. 1–9. IEEE (2014)
25. Xu, X., McGorry, R.W.: The validity of the first and second generation Microsoft Kinect for identifying joint center locations during static postures. Appl. Ergon. **49**, 47–54 (2015)