The E-Motion System: Motion Capture and Movement-based Biofeedback Game

D.Kelly*, D.Fitzgerald**, J.Foody*, D.Kumar**, T. Ward***, B.Caulfield** and C.Markham**

*Department of Computer Science, N.U.I Maynooth, Ireland.

**School of Physiotherapy & Performance Science, University College Dublin, Dublin, Ireland.

***Department of Engineering, N.U.I. Maynooth, Ireland

dan_kelly_ie@hotmail.com

Abstract:

This paper describes the development of a movement based training game aimed at teaching users an exercise program. This is achieved through analysing body posture as the player performs the exercise routine while concurrently receiving real-time feedback from the game. An in-depth post game feedback system also features, giving the player a detailed account of their performance after completing the exercise routine. Analysis of the player's posture is achieved by placing orientation sensors on appropriate parts of the players' body. The game can then read and interpret data from these sensors reconstructing a live 3D model of the players' posture. The game has the kinematic data of an expert performing the current exercise routine stored in memory, which is compared to the kinematic data of the current player and appropriate feedback is given to aid the player in performing the exercise. The theme of the prototype game currently developed is that of a yoga training game (E-Yoga).

Keywords: Motion Capture, Real time motion rendering, Biofeedback, Exercise Training, Kinematics, Performance feedback, Orientation Sensors,

I Background and Introduction

With the increasingly sedentary lifestyles of modern living, more and more people are suffering from various musculoskeletal pathologies such as back pain and neck pain. In addition obesity particularly amongst children is emerging as the most serious health challenge of our times. Ironically in the light of the contribution of this paper, computer and video games have been identified as one of the main culprits in contributing to the low level of exercise engaged in by children in the developed world. Therapeutic exercise programmes are advocated both to prevent and treat these physical conditions but adherence levels to such programmes are poor. Motivating people to participate in such exercise programmes is a challenge. We feel creating a computer game to increase the enjoyment during the exercise and give feedback and encouragement during the programme; players' motivations to participate in therapeutic exercise programs can be enhanced.

Current movement based games (i.e. EyeToy games) are based on 2 dimensional movements and allow the player to deviate from the desired exercise sequence without direct warning or feedback. This can be solved by tracking body movements using orientation sensors and analyze all 3 dimensions of the players' movement. We utilized commercial orientation sensors [1] to develop a prototype game with the view of integrating our own low cost sensors [2] at a later date.

The theme of the currently developed game is that of a yoga training game. Kinematic data of a yoga expert performing the sun salutation yoga exercise sequence was recorded by equipping a trained yoga teacher (i.e. the expert) with the orientation sensors and recording the kinematic data of the sequence (see Figure 1). The experts' data is stored in the system and a players' performance of the routine is calculated using a comparison system between the players' and experts' kinematic data.

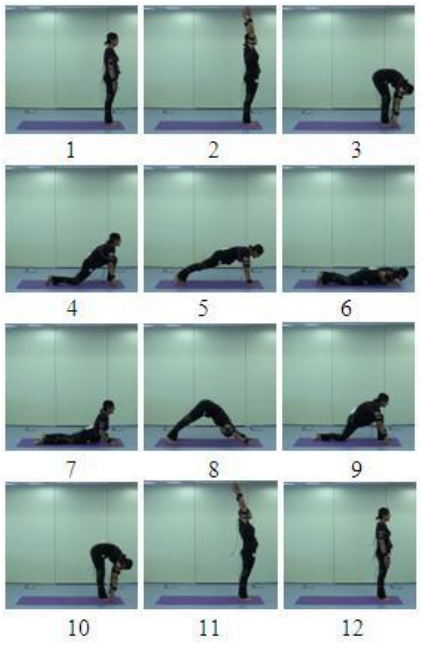


Figure 1 Expert training poses for Sun Salutation sequence.

The graphics and 3D environment of E-Yoga were designed with the aim giving the game a relaxing feel. The game was developed using the DirectX 9 API within a managed environment using C# as the development language.   The main contribution of this paper is the design and building of a real time kinematic feedback application capable of live motion rendering and feedback.

II Development

The development of the game can be divided into two components. (1) Motion capture development using orientation sensors and (2) development of the game framework, engine and graphics. Each of these two components will now be described:

Motion Capture Engine:

Ten orientation sensors are used as the basis of the kinematic sensor system for the development of the game. The Xsens Mtx sensor, seen in Figure 1, is the sensor used for the current version of the system. Each Xsens sensor is a small lightweight sensor which detects 3 dimensional orientation using 2 accelerometers, a magnetometer and a gyroscope [3].

Figure 2 MTx orientation tracker

Each of the ten sensors is connected to a small wearable base unit which sends data to the motion capture engine via Bluetooth. If each of the sensors is placed on appropriate parts of the body, the orientation of each part can be tracked dynamically. Using this data it is then possible to animate a 3D character model mimicking the movements of the player wearing the orientation sensors.

When modelling a 3D character for animation, the model is set up such that the bones of the character are in a hierarchical tree structure with the hips being the root of the tree [4]. This data structure means that a bones' position is described relative to its parent bone. However, each of the orientation devices detects its orientation relative to the global world meaning that some computation must be applied to the raw output of each sensor before the corresponding bone in the 3D model can be repositioned to a new orientation. The motion capture engine developed here does all orientation calculations using quaternion algebra [5]. To calculate the orientation of a bone relative to its parent bone the following equations are used:

$$q_{relative} \quad q_{raw} \quad q^{1}_{parentRaw} \qquad (1)$$

Where:

$$q^{\;1} \quad \frac{q_0 \quad iq_1 \quad jq_2 \quad kq_3}{q_0^2 \quad q_1^2 \quad q_2^2 \quad q_3^2} \qquad (2)$$

Where q_{raw} is the quaternion representing the global orientation of the current sensor and $q_{parentRaw}$ is the quaternion representing the global orientation of the parent sensor of the current sensor.

Using these equations we can now calculate a quaternion for each bone to describe its orientation relative its parent, and in the case of the root bone its orientation is defined by the corresponding sensors' raw output only as it has no parent. The game is developed using the DirectX 9 API, therefore all movements of objects within the environment must be defined as a rotation and translation using homogenous matrices [6]. To convert the quaternion calculated in (1) to a rotation matrix that can be used to rotate a particular bone within the game environment, (3) is used [5]:

$$M \quad \begin{array}{ccc} 2q_0^2 \quad 2q_1^2 \quad 1 & 2q_1q_2 \quad 2q_0q_3 & 2q_1q_3 \quad 2q_0q_2 \\ 2q_1q_2 \quad 2q_0q_3 & 2q_0^2 \quad 2q_2^2 \quad 1 & 2q_2q_3 \quad 2q_0q_1 \\ 2q_1q_3 \quad 2q_0q_2 & 2q_2q_3 \quad 2q_0q_1 & 2q_0^2 \quad 2q_3^2 \quad 1 \end{array} \tag{3}$$

In order for the 3D character model to accurately mimic the players' movements, the game must perform some calibration. The calibration consists of getting the player to stand upright with legs straight, arms parallel to the ground and looking straight. Then to calibrate, the sensors are reset having the effect of setting the current orientation of all sensors to zero degrees rotation about all axes, setting the origin pose for the player. All orientation changes made to the sensors, and thus the corresponding bone on the 3D model, will be made relative to the same origin pose. Therefore the 3D model now mimics the players' movement.

This motion capture engine is utilized within a game architecture to playback live and recorded motion, to record motion and to analyze and give feedback on motion. An overview of this game architecture is provided in the next section. Figure 4 shows the motion capture engine in operation within the game system.

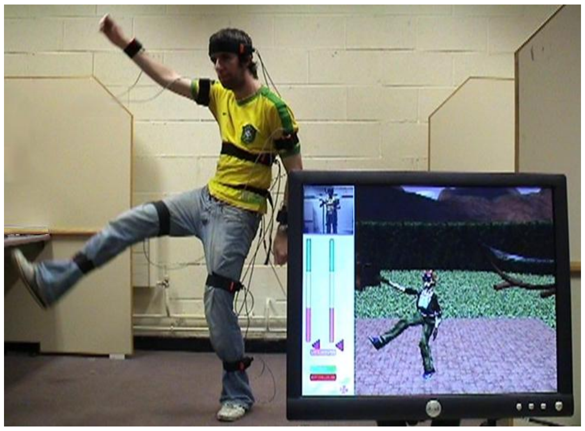


Figure 3 Real Time Motion Capture and Rendering

Game Engine / System Design

A game was developed to utilize the motion capture engine. The aim of the game is to teach players, wearing the motion sensors, an exercise routine by analyzing players movement and giving feedback. A screenshot of the game can be seen in Figure 3.

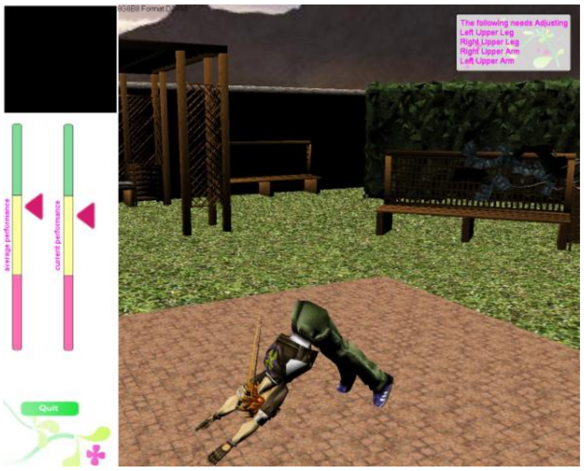


Figure 4 In game display; Expert in a Yoga pose (Pose 8: Downward facing dog pose)

The game system consists of the following main components:

1. Game Engine

This module manages creation of game modules, communication between game modules, input via mouse and keyboard, timing and rendering to the screen.

2. User Interface Modules

Modules which render user interface graphics and manage user input via buttons and text boxes.

3. In-game Controller

When in game mode the game can be in one of four different states at any given time:

i. Calibrate Mode

   The character model is animated using live Xsens data. A button is provided so users can invoke sensor calibration. After a successful calibration the 3D character model will mimic player movement.

ii. Expert Playback Mode

   The character model is animated using pre-recorded expert kinematic data and an audio description of each milestone pose is played at the beginning of each milestone sequence.

iii. Live Mode

   The character model is animated using pre-recorded expert kinematic data, while kinematic data for live player is being retrieved and stored in background. The experts exercise sequence contains a number of milestone poses which the player must perform before progressing (see Figure 1 for the 12 milestone poses for the sun salutation sequence). On completing each milestone pose, an audio playback encourages the player to the next pose. After completing the motion sequence, offline analysis is performed on the players' kinematic data. This is achieved by comparing data to that of the expert. Feedback on performance is then given and a detailed breakdown of the players' performance is displayed.

iv. Player Playback Mode

   The character model is animated using pre-recorded data recorded during the live mode i.e. playback of users motion.

4. Character Renderer

Creates a character 3D model from a specified .X file and animates the character given data from a specific instance of a Motion interface module (see below for Motion interface module description).

5. Motion Interface Module

This module provides kinematic data to the character renderer object from live Xsens sensors or pre-recorded Xsens data. A choice of 2 constructor overloads determines the source of the kinematic data. These two calls reflect the two possible modes of operation for the motion interface module. The first mode sets up the object so that data is retrieved from the set of live Xsens sensors and relative orientation calculations are performed on the data. The second mode sets up the object so that data is retrieved from a specified pre-recorded motion file. Regardless of the mode, data is retrieved by the calling object in the same manner, that is, a method is called with the input being an instance of a bone enumerator. The enumerator specifies which bones' kinematic data is to be returned. As a result, different instances of motion interface modules can be easily interchanged within the character render module, therefore changing between live animation and pre-recorded animation can be done in a transparent way.

6. Motion storage and control module

This module manages the loading and saving of kinematic data to and from file. When parsing from a file, kinematic data is stored in a 2D array with each row of the array corresponding to a single frame of animation. Associated with each frame of animation is a time, indicating when that frame should be used to animate the 3D character model, and a marker. Each motion sequence contains milestone postures which players must perform before progressing. The marker is used to indicate if the corresponding frame is or is not a milestone pose.

The motion object also controls frame timing. This is achieved by monitoring the amount of time elapsed since the last frame was returned to the motion interface module. Using this value it can calculate which frame to play next using the timing data associated with each frame. When in live mode and retrieving data from an expert sequence, the *get_frame* method requires an extra Boolean parameter specifying whether or not the player is in the same pose as the expert. If the expert is currently in a milestone pose then the time will not be advanced until such time as the player performs the same pose.

7. Environment Renderer

Manages loading and rendering of the 3D environment, lighting and camera position.

8. Player Info storage and control module

Manages storing, loading and creating player accounts.

9. Offline Feedback

After completing a motion sequence, the players' motion data is saved and loaded in a motion object. The offline feedback takes as input both the players and the experts motion object and calculates performance values for each of the sequences between milestones (4). Currently this figure is calculated using a distance metric in Euler space.

$$PoseSequenceRating \quad \overset{MileStone_{X\ 1}}{\underset{i\ \ Milestone_X}{}} \frac{\overset{NumBones}{\underset{j\ \ 0}{}} \left|P\ _{i,j}\quad E\ _{i,j}\right| \quad \left|P\ _{i,j}\quad E\ _{i,j}\right| \quad \left|P\ _{i,j}\quad E\ _{i,j}\right|}{NumBones \quad MileStone_{X\ \ 1} \quad MileStone_X}$$

(4)

Where E represents Expert, P represents Player, θ represents yaw , φ represents pitch and ψ represents roll

In addition a timing rating and a smoothness score is calculated for the sequence. The smoothness score is determined through differentiation and comparative analysis of the respective movement loci.

III Conclusion

In this paper we have demonstrated the possibility of a movement based biofeedback system allowing a player to learn, and get feedback on, an exercise routine such as a yoga routine. It has been shown that the posture of a person wearing orientation sensors, positioned at different parts of the body, can be modelled and analysed by the system. Modelling and analyzing a sequence of postures performed by an expert and an amateur (i.e. the player) and comparing the results can be used as the basis for a feedback system.

IV References

[1] www.xsens.com, Xsens Technologies B.V.

[2] J Foody, D Kelly, D Kumar, D Fitzgerald, B Caulfield, C Markham, T Ward, *A real time motion capture system, using usb based tri-axis magnetic and inertial sensors, for movement based relaxation,* Irish Systems and Signals Conference 2006

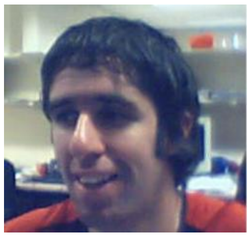
[3] MT Software Development Kit Documentation, Document MT0200P

[4] M. Meredith and S.Maddock, *Motion capture file formats explained*, Department of Computer Science, University of Sheffield.

[5] Jack B. Kuipers, *Quaternions and Rotation Sequences – A primer with applications to orbits aerospace and virtual reality,* Princeton University Press, ISBN 0-691-05872-5

[6] Tom Miller, *Managed Direct X 9, Graphics and Game Programming*, ISBN 0-672-32596-9, Sams Publishing

V Author Biography



Dan Kelly is a 23 year old Computer Scientist/ Software Engineer and has just graduated with a first class honours degree in Computer Science from N.U.I. Maynooth. Dan also has over 15 months experience working in the software industry working for Microsoft.