A Novel Approach to Sports Oriented Video Games with Real-Time Motion Streaming

Anton Bogdanovych
School of Computing, Engineering and Mathematics, MARCS Institute,
University of Western Sydney, NSW, Australia
A.Bogdanovych@uws.edu.au

Christopher Stanton
MARCS Institute
Bankstown Campus
University of Western Sydney, NSW, Australia
C.Stanton@uws.edu.au

ABSTRACT

We are currently observing a paradigm shift in virtual reality and simulation technologies. From being predominantly entertainment focused, these technologies are now finding a much wider use in the so-called "serious games" space, where playing a game teaches the player some useful skills applicable in the physical world. While this trend is booming in military simulations, training and even education, surprisingly enough, there is hardly any work available in the domain of sports. Most sports oriented video games do not teach skills that can be later reused in the physical game environment and thus there are minimal benefits to the player's "real-world" performance. Performing key sports actions such as shooting a basketball or hitting a tennis ball are normally done via actions (like pressing keyboard buttons or swinging a game controller) that have little correspondence to the movement that needs to be performed in the game's physical environment. In this paper we advocate a new era where it is possible to play simulated sports games that are not only enjoyable and fun, but can also improve the athletic skills required for the real-world performance of that sport. We illustrate the possibility of this idea via state of the art inertial motion capture equipment. To highlight the key aspects of our approach we have developed a basketball video game where a player pantomimes dribbling and shooting in a virtual world with a virtual ball. Importantly, the virtual world of the game responds to a player's motions by simulating the complex physical interactions that occur during a physical game of basketball. For example, if a player attempts to score a basket, the simulated ball will leave the player's hand at the appropriate time with realistic force and velocity, as determined by motion capture and the physics system of the selected game platform. We explain how this game was developed and discuss technical details and obtained results.

INTRODUCTION

During the last decade the gaming industry has experienced a significant change of paradigm. Traditional input devices like keyboard, mouse, touch screen and the hand-held game consoles are being widely replaced by the more advanced technology like Nintendo Wii [15], Playstation Move [17] or Microsoft Kinect [13]. Players are embracing the change that allows them to employ their entire bodies in the game and interact with the computer in a more natural way. With these new technological advances the players are becoming more active and playing in this way is believed to be beneficial for weight management [8], which is rapidly becoming a significant problem among young gamers.

Despite recent advances in human-computer interaction, most video games existing today do not fully utilise the motion capture potential of modern equipment. One of the key reasons for this is that the majority of technological solutions used in today's video games are incapable of error-free real-time motion capture and, thus, must rely on gesture recognition and custom animations rather than real-time motion streaming. For example, systems like Nintendo Wii and Playstation Move normally utilise just one or two sensors, so they can only be used for tracking the hands of the user, and are incapable of correctly estimating the rest of the body joints. Camera-based solutions like Microsoft Kinect focus on movement recognition for the entire player's body and depend on sophisticated body detection algorithms, but such sensors are still producing too much error, are prone to sensor occlusion problems and only deliver satisfactory results when multiple sensors are combined (which in itself involves a range of engineering problems related to sensor synchronisation and projection occlusion problems) [2]. While most existing markerless motion capture solutions are not capable of accurate and error free real-time motion capture, there is hope that one day they will get to the stage when accurate real-time motion capture with cheap systems like Microsoft Kinect becomes an every day reality. This hope is supported by some experiments showing relatively small error for Microsoft Kinect in a controlled environment [6] and the authors also suggest that having multiple Kinects would improve the situation. In regards to accurate real-time motion capture, one the most reliable approaches at present is to employ full-body inertial motion capture suits (such as XSENS MVN [1]. Such equipment is capable of highly accurate real-time motion capture suitable for direct streaming onto a video game avatar with little noticeable detection error.

Due to significant errors, most current game solutions that employ motion capture have to rely on mapping the captured

EICS'15, June 23 - 26, 2015, Duisburg, Germany.

DOI http://dx.doi.org/10.1145/2774225.2774836

player motion to a pre-recorded motion sequence from a motion collection. Thus, what the player observes on the screen is not the motion performed by the player, but a somewhat similar motion from a motion repository. Very often this is a "perfect" recording of a professional athlete performing the corresponding movement. With inertial motion capture, however, it becomes possible to stream the actual player motion onto a game avatar in real time, allowing the player to see his/her actual movement with all its flows and imperfections. In order to integrate this motion streaming with the game world and enable player interactions with game objects, all that is required - is to map the physics system of the physical world to the physics of the virtual world, so that player actions correctly change the state of the video game, trigger desired animations and apply necessary physical forces to the objects the avatar is colliding with.

This allows for a novel motion streaming paradigm to be applied to video game development. This paradigm allows the player to interact with the game world by performing the kind of movement that is very similar to what would have been performed in the physical game environment. So, the player is able to see his/her own motion, learn from mistakes being made and even receive automated improvement suggestions. It also becomes possible to fine-tune the system so that virtual world of the game and its physics system closely match with that of the physical game space. Establishing such a connection allows for treating the video game as a potent practice environment suitable for improving the skills that are directly applicable in the physical world.

From game developers' point of view, motion streaming may provide a simplified universal approach to game development for a dedicated cluster of sports-oriented motion capture games. Once a universal way of streaming the user motion onto an avatar is established, building a new game can be as simple as creating the corresponding objects the player must interact with (e.g. soccer ball) and supply these objects with scripts capable of correctly interpreting the user motion and animate this objects in response to user actions. So, creating a new game becomes a matter of designing a new object and attaching a script that correctly animates this object in response to user movement.

To illustrate the idea of motion streaming focused game development, we have created the Motion Capture Basketball video game. In this game the player, equipped with a XSENS MVN suit, pantomimes various basketball moves, while the corresponding avatar is replicating those moves and plays with the ball in the virtual world of Second Life. The key novelty shown in this game is that there is no gesture or motif recognition being conducted. Instead, the data received from the suit is being processed and streamed directly into the virtual world. This data stream is then used for animating the avatar and estimating the ball movement based on the physical parameters of the player's body joints. The motion stream is broadcasted to all the objects present in the virtual world. So, what becomes possible is to have objects encapsulating key game mechanics. While the game engine is responsible for standard motion processing, animating the avatar and keeping track of physics, the objects in the virtual world will encapsulate the scripts responsible for correct processing of the motion stream and animating avatar interactions with these objects accordingly. Thus, for making a soccer game, for example, it is only required to create the soccer ball object and change the court accordingly. Then the ball's script must be modified so that it uses the motion data of the feet rather than hands and new algorithms must be created for detecting when the ball must be attached or detached from the player. Creating such objects by external parties is flexibly supported in a virtual world of Second Life, which has a well established market place and a healthy economy to support it. So, a soccer game can be offered in Second Life by means of selling the soccer field object, which would include all the necessary scripts available within this object.

RELATED WORK

The vast majority of modern players are still interacting with their video games through traditional hand-held devices, joysticks and keyboards. In order to perform various actions (like throwing a ball or kicking an opponent) they must normally press a combination of buttons that map to a certain predefined movement of the player's avatar or some other game action. A few modern gaming solutions that try to make the gaming experience more life-like have started to appear during the last decade. Below we present an overview of some key technologies with a particular focus on consoles and sensors that use player movement as an input.

Nintendo Wii

Nintendo Wii is one of the most popular of such modern game consoles. In its traditional setup Wii relies on the Wiimote controller that uses an embedded accelerometer for tracking the position and rotation of the device. By holding a controller and moving it around, a player transmits position and rotation data into the system. This information is then used for controlling the avatar of the user and objects in the game (e.g. a tennis ball). The most common approach to interpreting the controller data and then linking it to certain actions in the virtual world of the game is gesture recognition. A certain motion pattern performed by a user is first filtered and then matched against a number of predefined "gestures" [15]. When a gesture is recognised - the corresponding virtual world action is triggered and the player's avatar is animated accordingly. The precision of the Wii hardware is not ideal and the error rate in gesture recognition lies somewhere between 10% and 20% [15]. To address this issue, game developers normally allow for quite a wide range of motion when detecting the type of movement being performed. As the result, a movement a player performs for triggering a particular avatar behaviour only slightly resembles the actual resulting animation being played. For example, hard serves in tennis can be produced by just slight movements of the wrist that would have resulted a completely different ball movement if performed with a tennis racket on the actual physical court.

Playstation Move

Playstation Move is a more recent and a more precise alternative to Nintendo Wii. It also comes in a form of a handheld

device and uses a video camera to track the device position in combination with inertial sensors for detecting rotation and its overall movement [17]. Similar to Wii - this platform is quite successful with games like tennis, baseball, cricket or golf, where the game controller itself resembles the instrument with which the ball is being hit. However, it is difficult to use Playstation Move with games like soccer or basketball, where the ball is thrown or hit by the player (rather than using additional instruments). The resulting avatar animation in such games normally involves the movement of all body parts, but the game controller is only supplied with one or two sensors. Thus, there is not enough data provided by the hardware and game developers must rely on gesture or motif recognition for determining which animation to trigger.

Microsoft Kinect

One of the most promising technologies that is not limited to tracking only one or two body joints is Microsoft Kinect [13]. It was released in late 2010 and features a static sensor that is normally installed on a desk or on top of a television set. The player is not required to hold any control devices and interacts with the sensor via natural movement. The combination of a video camera and an infra-red depth sensor helps to estimate the positions and rotations of various body parts of the user and utilise this information for controlling the game experience [13]. Kinect relies on computer vision and employs similar algorithms as other markerless camcoder-based motion capture and human body recognition methods.

A number of limitations of existing camcoder based motion capture methods were listed in [19]. Even one of the most advanced motion capture techniques that uses multiple camera views [21] is not only prone to occlusion problems, but is also incapable of fully automatic motion capture as it is reported to require manual pose correction for 13 out of 250 frames [21] . Furthermore, markerless mocap algorithms often dramatically fail if the subject wears loose clothing [7], although in other situation markerless mocap solutions like the latest Kinect sensor can show similar recognition performance to that of marker-based systems [4]. Relying on the existing body of literature and based on our own interaction experience with Kinect we can conclude that due to a relatively low resolution of the camera and only a single view (although together with depth sensor) being used - it is not currently feasible to capture error-free real time motion of the user. Similar to Wii and Move, instead of streaming live motion onto an avatar Kinect based games normally rely on gesture (or motif) recognition. The closest work available in the literature that is similar to Kinect's motion capture method is described in [3]. It presents a technique of dynamically recognising various player gestures and blending in the recognised gestures and the 3D model of the avatar. The authors report that their approach is prone to latency issues and errors in recognition, which is consistent with our observations.

This technology, however has a lot of potential and recognition errors will most certainly be solved in the near future to a large extent. While the occlusion problem will be difficult to solve, it can be minimised by employing a number of synchronised sensors as suggested in [2]. Next, we present an overview of modern approaches to using motion capture as input for controlling game characters.

Motion Capture in Gaming

One of the most interesting attempts to employ interactive motion capture in the gaming domain is [10]. This work relies on a pre-recorded motion database and uses the input of the motion capture equipment for identifying the matching motion in their collection. Once such motion is found - it is modified to fit in the game space, so that the constraints and affordances of the objects there are satisfied.

A similar approach is taken by [11], where a pre-recorded motion collection is also employed for detecting the most suitable player motion. The virtual environment and the objects placed there are also used to enforce "kinematic constraints" on possible movement types. The key emphasis of this work is on helping animators to simplify producing impressive character movement in response to interactions with virtual objects, so that key emphasis is on developing techniques for motion editing in response to modified object movement trajectories rather than on using motion capture for player movement and estimating the behaviour of surrounding objects.

One of the most detailed descriptions of interactively matching a currently performed player movement with the most suitable element from a pre-recorded motion collection is presented in [5], who demonstrate how an inexpensive technological setup, featuring 2 video cameras and players attaching a few reflective dots to their closing, can be successfully utilised for motion sequence detection.

A more recent take on the problem of motion detection is presented in [12]. A collection of motion recordings is also employed, but rather than recognising a motion sequence, the authors propose to recognise individual poses. The equipment being used in this work is the same as proposed in this paper (XSENS MVN [1]). The work of [16] also investigates interactive motion detection, and presents a comparison of player experience using traditional joystick based game controllers versus Nintendo Wii controllers for interacting with a simple robotic character. The results speak in favour of more natural ways of interaction using Wii.

An interesting overview of various technologies for interactive control of video game characters is presented in [14]. This essay has been conducted under supervision of XSENS technical staff, so it is particularly interesting in disclosing some of the limitations of the modern motion capture equipment, including the XSENS MVN suit (which we are using). This knowledge is particularly useful as the XSENS manufacturers rarely reveal the technological details behind their equipment.

In summary, most existing motion capture focused approaches in interactive gaming rely on gesture or motif recognition, mainly due to limitations of the motion capture equipment being employed. We did not find any evidence of real-time motion streaming being employed in any of the games. Below we further explain the details of this technique and showcase the basketball game that was developed following

this principle. While the game requires some rather expensive and bulky equipment, which appears to be not very practical, we would like to stress that our aim is not to promote inertial motion capture as the way of the future and suggest mass production of motion streaming oriented games employing XSENS MVN. Instead, we use inertial motion capture as an intermediate research step while waiting for less expensive and less intrusive technologies that have a comparable degree of accuracy.

Another point we would like to clarify is that not all players would necessarily enjoy seeing their true motion instead of much better looking motion recorded by professional athletes. Also, many players would not want to have a full-body basketball exercise and would be more interested in games where they do not have to get up from the couch and can play virtual basketball using a standard game console instead of pantomiming basketball shots. However, similar to dynamic Kinect-based games, there is a market for basketball players who want to enter a more realistic experience that could potentially offer improving their real-world basketball skills.

REAL-TIME MOTION STREAMING FOR GAMES

We see the idea of real-time motion streaming as the key contribution of this work, where the proposed concept is to stream real-time motion, captured from the player, directly onto the player's avatar and enabling the surrounding environmental objects to realistically react to this stream, as expected in the relevant sporting domain. In this way animating the avatar becomes a standard feature of game engines supporting real-time motion streaming and is no longer a concern of game developers using the engine, while game developers spend more of their efforts on creating algorithms associated with objects that need to be animated in response to listening and parsing the motion stream generated by the player.

As it was mentioned earlier, traditional motion capture approaches predominantly rely on creating pre-defined motion sequences and recognising those in the movement the player performs. These approaches are prone to recognition errors and latency problems, however, they are still quite suitable for entertainment purposes of the majority of video games were high precision and realism are not critical. In our work, however, we investigate an entirely different method that is not currently suitable for mass market production, but might be extremely beneficial for professional athletes or sport enthusiasts interested in improving their real-world skills.

We propose to employ a high precision full-body motion capture suit as a game console. Only recently motion capture suits similar to Xsens MVN reached the level of precision when they can correctly capture real-time motion of a human body with no significant data errors. XSENS MVN is capable of real-time motion capture with very high accuracy (0.8°, $s = 0.6^\circ$ for each of the sensors) [19]. Using such precise equipment allows for direct real-time motion streaming from the player body onto a 3D model of the video game avatar. Thus, rather than relying on gesture recognition, we can animate the avatar with the actual player motion and make the adjustments in the virtual world of the game to reflect on the actions those motions may have caused.

Motion Capture Suit

XSENS MVN comes in a form of a Lycra suit with 17 embedded motion sensors. The suit is supplied with MVN Studio software that processes raw sensor data and corrects it. MVN studio is capable of sending real-time motion capture data of 23 body segments using the UDP protocol with the frequency of up to 120 motion frames per second. The key elements of the data being transmitted are absolute (X,Y,Z) position of each segment and its absolute (X,Y,Z) rotation.

Architecture

A high level overview of the proposed system architecture for real-time motion streaming is shown in Figure 1. The motion capture suit transfers via wireless communication sensor data (positions and rotations of 23 body segments) to the game engine. The game engine translates this motion into the corresponding avatar animation, but at the same time it also passes on the motion data to all objects that have been registered as motion listeners. Every such object (in our case O1 - is the ball and O2 - is the basket) would have a corresponding motion listener script attached to it and the script is responsible for making sense out of the motion data and then using it for making the corresponding object move on the screen as the reaction to the motion stream generated by the player.

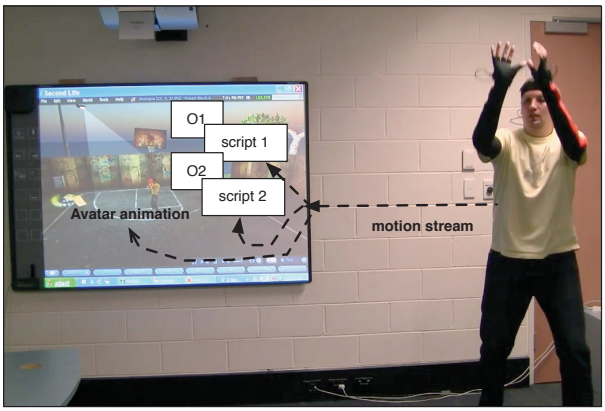

Figure 1. System architecture

Not only Xsens MVN, but any motion capture solution capable of streaming real-time motion data in the form of position and rotation data for every body joint would be suitable for participating in the system. The majority of existing motion capture solutions provide a data stream, from which it is possible to obtain the resulting positions and rotations of the corresponding body joints. So, while an approach of extracting such data frames is normally hardware specific, the rest of the implementation is hardware independent.

Integration with the Game Engine

To be able to test our assumption about the possibility of using Xsens MVN for gaming purposes, we selected one of the freely available gaming platforms (Second Life) that also comes as an open source version. Second Life supports multi-user interactions, features a highly advanced Havoc physics

engine and supports dynamic object building and complex actions with the objects in the game. To enable the suit connectivity we modified the game client so that it receives real-time motion data from the motion capture suit, processes it, broadcasts it in a compact form to all surrounding objects and correctly maps this motion data to the body joints of the player's avatar. In this way we have created a generic platform suitable for rapid development of motion capture games.

From a game developer's perspective, there is no need to modify the motion data transmission algorithms and adapt them for every game, as those algorithms are universal. A game player can simply wear the suit and perform basketball, soccer, boxing or tennis moves. To make it become a game - the corresponding objects must be created and programmed to handle player movements accordingly. Programming those objects does not even require having a motion capture suit and can be done by any third-party developer.

Connecting XSENS MVN to an avatar

In order to animate the avatar in the virtual world in real time, we directly map each body segment of the human user to the corresponding avatar joint and adjust those accordingly. The data being transmitted by the suit is measured using the absolute coordinate system of the suit space, where the X axis is aligned with the magnetic north and the origin corresponds to the position at which the suit was turned on.

The avatar we animate is positioned within the coordinate system of the given Second Life region, the rotation component determining the body orientation of the avatar will be measured using this coordinate system and the body proportions of this avatar are normally different from the body proportions of the human wearing the motion capture suit. Thus, for controlling the movement of the avatar we need to continuously translate the positions and rotations of every joint into the space of Second Life and apply scaling factors to compensate for the difference in body proportions.

The location of the player in Second Life is determined by the location of the player's chest joint. Moving the player around the virtual world corresponds to updating the location of the chest joint. Therefore, to correctly translate the player movement in the suit to the movement within the virtual world we need to establish a connection between the chest sensor of the suit and the chest joint of the Second Life avatar. To establish such a connection we applied standard coordinate transformation that translates the displacement of the chest sensor from the origin of the suit space to the coordinate system of the game space.

Equipment Accuracy

To test the accuracy of the motion capture suit and verify wether it is suitable for our purposes we have tested XSENS MVN on the basketball court[1]. The player wearing the suit conducted a 20 minute practice session exercising a variety of dribbling and shooting moves. The training session was recorded on camera and the real-time movement was captured and stored on a portable laptop computer.

Figure 2 shows a side-by-side comparison of the camera recording of the player shooting the ball and the resulting 3D skeleton captured by XSENS MVN.

Figure 2. Testing Equipment Accuracy on the Court.

The captured motion sequence was highly accurate. Apart from a couple of slight miscalculations of the hand (probably due to very high movement speed during the shot and the hand sensor coming off as the result of this) the resulting data was very clean. The miscalculations were not significant and not immediately visible with a naked eye, but were spotted after analysing the recorded sequence in slow motion.

MOTION CAPTURE BASKETBALL

Once we have finalised our solution for transforming the virtual world of Second Life into a motion capture gaming platform - we developed the basketball game to verify the capabilities of this platform. Figure 1 shows a player pantomiming the turn-around jump shot, while his avatar mimics all his movements and interacts with the ball[2].

In our game setup the player must wear Xsens MVN full-body motion capture suit while standing in front of a big screen that visualises the 3rd person view of the player's avatar in the virtual world of the game. The player's avatar starts the game by mimicking the pose of the player and is initially assumed to be holding a ball. To interact with the ball the player must pantomime shooting and dribbling movement as if interacting with a ball in the physical world. The virtual ball that is visualised on the screen is a physical object that obeys the rules of physics of the Second Life platform. To make the game experience as close to the physical game as possible, the virtual ball was designed in accordance with official size and weight standards of the NBA [18].

Motion Capture Equipment Calibration

It is expected that Xsens MVN inertial suit is to be used by various people with different body proportions. Even when used by the same person, it is impossible to wear the suit so that all sensors perfectly align and appear in the same position every time. Therefore, a short calibration process is required before every use. Depending on the desired quality of the resulting correspondence between the bodies of the virtual character and the person wearing the suit, the calibration

[1]The video showing a side-by-side comparison of the video stream and raw motion data for a part of this training session can be seen at http://www.youtube.com/watch?v=w3dbMml8BOI

[2]The corresponding video for this game is available online at: https://www.youtube.com/watch?v=rCqy8GuqfAU

process can take from 20 seconds to 5 minutes. The simplest mode of calibration involves mimicking a number of poses as shown in Figure 3. From our observations, simply mimicking the neutral pose shown in the picture is enough to establish a relatively high level of correspondence.

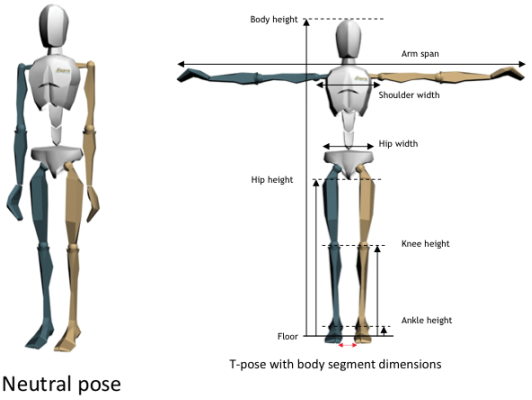

Figure 3. Key Poses to Replicate for Equipment Calibration

If more precision is required, it is necessary to enter numeric values for player dimensions (see the right hand side of Figure 3) and additional poses and movements must be performed by the player. Once the calibration is completed - the player can start the game and is no longer required to calibrate the equipment while the motion capture suit is on.

Animating the Avatar

In order to animate the avatar in Second Life we have modified the Second Life client software, so that it can receive the motion capture data from MVN studio in real-time. Once the motion data is received - the incoming absolute positions and rotations are translated into relative coordinates of the given avatar. On top of those transformation we also apply scaling and coordinate transformations, so that the movement can be correctly visualised with the current avatar body parameters. Once all the transformations are completed - the avatar is animated by applying the motion data to each of the body segments. To make it possible for other players to see the animation - the motion data is propagated to all the connected clients, where the given avatar is also animated.

At present, the position of the player on the field is determined by the relative position of the chest sensor. So the player can walk and run around the court by simply moving around the room where the game is being played. This applies a limitation on the room size for playing the game, as the room where the game is played must be at least $14m^2$ large to cover a half of the basketball court.

Holding the Ball

Once the player enters the game world - the ball appears in the player's hands. Further behaviour of the ball is driven by the actions of the player. We consider that the player is holding the ball if the ball is positioned in-between two hands and the distance between hands is approximately equal to the diameter of the ball. If this distance is greater - the ball immediately detaches from the player hands and falls on the ground under the influence of the gravity force, otherwise - the ball remains attached to the player's hands and moves alongs whit them. The above information is dynamically obtained from the data being transmitted by the hand sensors.

Dribbling

Dribbling behaviour starts when the ball collides with a hand of the player. If the ball is on the ground at the moment when the dribbling starts - we apply an impulse that is calculated using the acceleration of the player hand to the ball in the direction of the hand movement. The physics system of Second Life is then responsible for animating the movement of the ball, making it bounce off the ground and return back under the influence of gravity. When the ball is in the air - the player can change its movement trajectory in a similar way by making one of the hands collide with the ball. When such a collision is detected - an additional impulse is applied to the ball as described above. We also consider situations when the player is holding the ball with one hand while standing or dribbling, so we make a simplistic assumption that the ball should attach to the hand and move together with it if the relative rotation angle of the hand with the ball in relation to the floor is less then 45 degrees.

Shot Detection

The movement of the player hands is also tracked for detecting the moment when the ball is being shot and for determining the direction and velocity of the ball at this moment. Based on the analysis of a number of training session motion recordings we conducted on the field, we came up with the following simple heuristic for shot detection. We assume that the shot is fired if the following conditions are met:

1. The player is holding the ball
2. The ball is above the player's chest
3. The distance between the hands becomes larger than the diameter of the ball

Provided that a proper mapping is established between the physics system of the virtual world and the physics of the real world, a shot that results scoring in Second Life would also result scoring on the physical basketball court.

Shot Detection - Machine Learning

The use of the heuristic described above for detecting the precise moment at which a ball is released from a player's control while shooting was satisfactory in many situations, but not all. To deal with those situations, an alternative approach was employed which involved the use of supervised machine learning to determine when a player is or is not in "possession" of the virtual basketball. The final system was using the combination of the simple heuristic and machine learning. Whichever method first detected the shot triggered the ball release.

Shooting the Ball

In order to create the illusion for the player that he/she actually plays basketball, it is essential to simulate a virtual ball

that will move in response to player movements. For simplicity, we have decided to rely on the underlying physics engine of the Second Life platform to simulate ball behaviour. The physics engine provides a possibility to specify the mass of the ball, which was set to be the same as the official size of the basketball (567 gm) as prescribed by the National Basketball Association (NBA) [18]. Once the mass of the ball is set - we can directly call a method of the physics engine (llApplyImpulse()), which given the vector of the force applied to the ball will apply the desired impulse and will take care of the resulting projectile motion of the ball, detection of collisions with other objects, gravity and other physics elements. This significantly simplifies the task of ball movement visualisaion, as all that is required is to compute the force applied to the ball in the moment of shooting.

In order to calculate the force applied to the ball we rely on the second Newtonian law of physics that can be summarised in the following equation;

$$F = m * a \tag{1}$$

Here F - is the force that is required to be passed to llApplyImpulse() method, m - is the mass of the ball and a - is the acceleration of the ball in the moment when the ball is being released from the player's hands. Detecting this moment (shot detection) was explained above.

For simplicity, we assume that the player is shooting the ball with his/her right hand and the left hand and the acceleration of the ball in the moment of shooting is equal to the acceleration of the player's right hand in this moment. So, computing acceleration of the ball can be done by using the change in position of the corresponding sensor of the motion capture suit. Let us assume that the player's right hand position is the following coordinate in the 3-dimensional space:

$$P =< P_x, P_y, P_z > \tag{2}$$

As known from physics, instantaneous velocity is a derivative of position:

$$V(x) = \lim_{\triangle t \to 0} \frac{P(t + \triangle t) - P(t)}{\triangle t} = \frac{d}{dt} P(t) \tag{3}$$

Acceleration is known to be the first time derivative of velocity, or a second time derivative of position:

$$a(t) = \frac{d}{dt} V(t) = \frac{d^2}{dt^2} P(t) \tag{4}$$

The motion capture suit provides position updates of the each sensor every 1/120 of a second. So to compute the velocity and acceleration we use position changes of the right hand sensor for every such snapshot of data. One of the methods to compute the acceleration relying on equations 2, 3, 4, is to employ Euler method [9]. Given that we know the position changes every 1/120 sec and the time difference between those positions is very small, we can use those positions as a basis for polygonal approximation of the velocity and calculate a local velocity for each of the known position. Once we have those local velocities, we can then apply Euler method again to compute local acceleration. The acceleration we need is the one that corresponds with the position of right hand at the moment when the ball is being released from player's hands.

Multiplayer Aspect

Second Life is a multiuser virtual world, where the ability of users to jointly use objects and see one another in the virtual world is supported by the original platform. For our game this means that it is possible for people to play together. In this case every player must wear a motion capture suit.

Player Learning Aspect

Our game provides a unique opportunity for players to observe themselves in the game and correct their playing style if necessary. Second Life offers facilities for creating custom camera views, recording player motions, letting a player to zoom a camera to see the up-close details of the certain body parts. It is also possible to show super-slow-motion replays. Furthermore, through the multiuser aspect of the game a player can seek professional help by inviting a coach into the game and letting the coach observe the movement and correct it by giving an advice via chat or voice facilities provided by Second Life (or simply by showing how a certain motion must be correctly performed). The player learning aspect was further explored in one of the case studies presented below.

DISCUSSION OF RESULTS

The initial intent of our work was to closely replicate a basketball field in the virtual environment and fine-tune the system so that the reconstructed virtual court precisely matches the physical court and so that the physics of the game closely matches with the physics of the real world court. Our initial idea about validation experiments was to compare the player wearing the motion capture suit and playing with a physical ball on the court with the movement of the player and the resulting ball behaviour in the virtual world. However, while building the system we have discovered some limitations of the motion capture equipment. One of the key limitations was the inaccuracy of position tracking. As noted by [14] inertial motion capture technology is very accurate in measuring rotational data and is not so well suited for measuring position changes. Through our experiments we discovered that player's position changes on the court seem to be computed based on kinematic reconstruction of leg movement and employing some heuristics about a foot touching the ground and advancing the position of the person as the result of this. For the case of basketball, however, where players often jump and move their feet while in the air, those heuristics seem to produce errors. So, we found it to be impossible to have the identical experience for the player in both physical and virtual environments; such close correspondence seems only possible with employment of additional equipment capable of accurate position tracking. After giving up on the idea of mimicking the physical ball/player behaviour in the virtual world, we had to change the experiments to focus on user experience. In this respect, we have conducted 2 case studies aiming to test the following two hypotheses:

- **H1**: Using motion streaming allows professional basketball players to improve their skills.

Figure 4. a) Dribbling the Ball b) Long Range Shot c) Turn-Around Jump Shot

- **H2**: Using motion streaming allows novice players to learn new game skills.

Below we present two case studies each testing one of the aforementioned hypotheses. Due high equipment cost and relatively long setup time (about 20 minutes for each participant), it was difficult to test those hypotheses with a statistically significant sample of users. So, instead, we have conducted extensive experiments with 2 participants. After those experiments it was obvious that our hypotheses are correct and further studies would not change the outcome.

Case Study H1: Skill Improvement

In order to obtain player impression and understand how the game can impact the players, a member of our team, former semi-professional basketball player who participated in various competitions of the university basketball league was asked to evaluate our game. The evaluator spent over 10 hours playing the game and then shared his impressions. It was noted that the lack of tactile feedback caused some initial problems, but the player quickly adjusted and stopped noticing it, as the visualisation of the ball was able to compensate for the lack of sensing it through touch. This ability of our brain to compensate for missing sensors and replace those by other types of sensors is well documented by neuroscientists [20]. Our preliminary experiments confirm this thesis.

The overall impression was very positive and the evaluator noted that he was able to quickly adjust to the game and make good progress. One of the key problems that posed some difficulties is the inability to correctly identify the position of the basket based on a single 2D image on the screen. The evaluator noted that it was more difficult to score in the game than on the physical court, because the depth perception was different.

Overall the evaluator was impressed with the high realism of the game and commented that he has noticed an improvement in the shooting skills on the actual physical court after playing the game. The participant said that the experience of spending so many hours shooting the virtual ball in the game and then going to play on the physical court was comparable with the experience of shooting the ball on a court with non-standard measurements and basket height. In both situations adjusting to the new court required some time, but the skill of getting the ball in that was practiced in the different environment started to show after the adjustment was learned.

The participant has also mentioned that the interaction with the ball was natural and highly realistic. There was no need to learn any new controls and the resulting ball behaviour was quite predictable and very similar to what the ball behaviour would have been on the physical basketball court. The ball bouncing was noted to be much harder, as if the ball was heavily over-pumped. The court dimensions seemed a little smaller than on the physical court (even-though they were precisely measured and were identical), which may be because of the lack of the depth perception. The player was quite fascinated by the possibility to see his own movement on the field and commented that it would have been a very helpful feature for people who are still learning the game.

Case Study H2: New Skill Acquisition

In the second case study we investigated the possibility of using the game for improving the basketball skills of a novice player. For this task we have selected a participant with little prior exposure to basketball. The essence of the experiment then was to see whether by interacting with the game world this person would be able to learn new basketball skills. In the future we intend to develop a repository of moves and shots and use this repository as the basis for detecting the movement the player is trying to perform and highlight the key differences between the current movement and the "ideal" movement. However, in this work we only intend to show that in fact such improvement suggestions can be made and can actually lead to an improved on court performance. Therefore, we designed a simplified experiment where the the test subject was interacting with the system via the motion capture suit and another experimenter observed the players movement on a computer screen and advised the player on possible improvements via a microphone.

Prior to the experiment in the game the player was taken to the physical court and asked to perform simple dribbling and shooting (to establish the initial skill level). After this, the player was asked to wear the motion capture suit and interact with the game environment, while listening to the advice of the basketball consultant.

The study showed that the basketball consultant was able to successfully recognise the intended movement of the player and give advise on improving this movement. Both shooting movement and dribbling movement were performed by the player and as the result of voice suggestions that player was able to improve those skills.

After completing the experiments in the game the player was then taken to the physical court to see whether the acquired skills are applicable in the physical world. Both shooting and

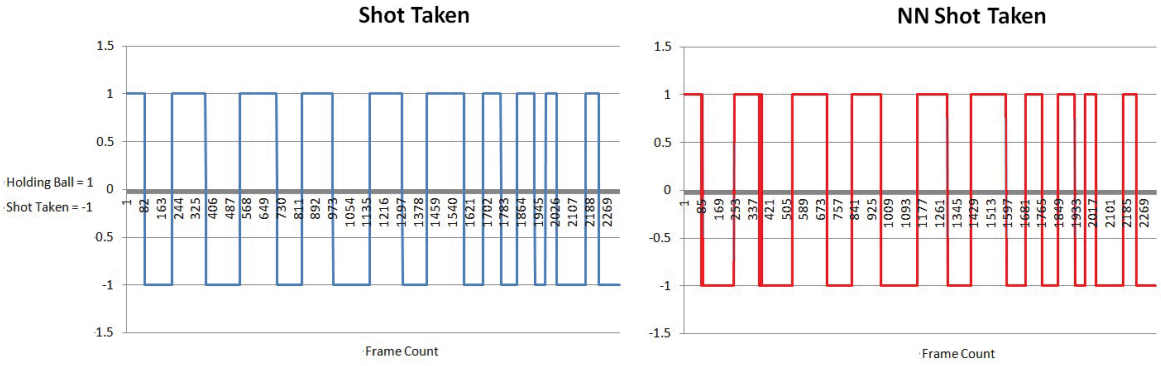

Figure 5. Machine Learning Training Data. The training data (left) for 10 basketball shots, where the value 1 represents holding the ball, and the value -1 represents releasing the ball. The trained neural network is illustrated (right).

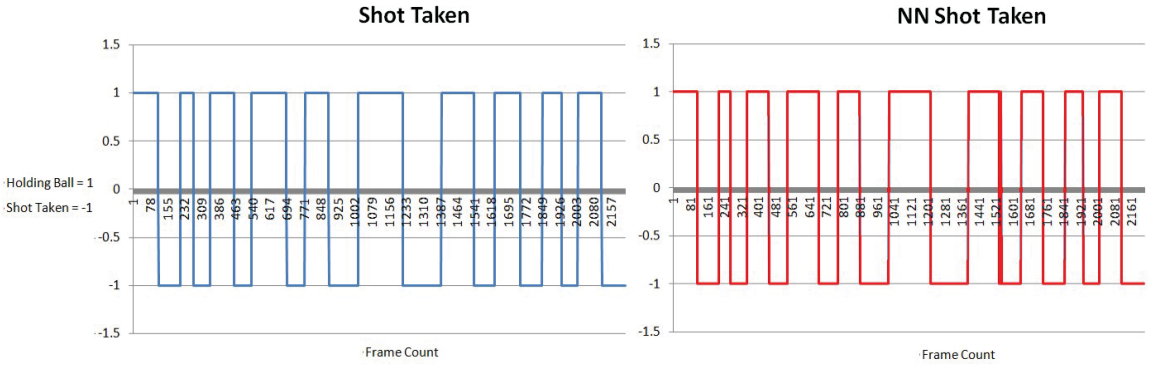

Figure 6. Machine Learning Testing Data. The testing data (left) for 10 basketball shots, where the value 1 represents holding the ball, and the value -1 represents releasing the ball. The trained neural network is illustrated (right).

dribbling have improved. The player has also commented positively about the possibility of seeing her own movement and mentioned that it seemed to have a significant positive effect on learning.

Shot Detection Experiments

Another aspect that was important to evaluate is the accuracy of shot detection. The heuristic that was used always produced a correct result when the ball was being shot in a way that satisfies this heuristic, but in other situations we had to rely on machine learning for shot detection. Figure 5 shows the neural network (NN) learning result for 10 shots. The NN was able to closely approximate the timing of ball release. Figure 6 shows the NN performance for 10 novel shots for which the neural network was not trained.

A total of 20 example basketball shots were recorded in the field. For each of these 20 shots, the motion capture data was manually annotated to reflect whether at that point of time, the player had or had not released control of the ball.

A feed forward neural network with one hidden layer of 30 neurons was trained to approximate a function which predicts binary output of whether ball has been released *(r)* based upon the distance between the player's hands *(d)*, the acceleration of the player's hands *(a1, a2)*, the velocity of the player's hands *(v1, v2)*, and the set of quaternions hand rotations *(LQot, RQot)* captured by the motion capture suit, i.e. $r = f(d, a1, a2, v1, v2, LQot, RQot)$. The function is represented by a matrix of values which represents the weights of each node on each other. These weights are optimised to approximate the function (training) as closely as possible using particle swarm optimisation. Once the neural network has been sufficiently trained, we were able to employ it for novel movements to detect whether the player has or has not released the ball.

Analysis of the error rates for the novel set of 10 basketball shots reveals that of a total of 2222 frames of motion capture, the neural network erred on detecting whether the player has or does not have the ball on 38 occasions (an error rate of 1.7%). Interestingly, only on 2 occasions did the neural network incorrectly infer the player had the ball when they did not, but on 36 occasions the neural network incorrectly inferred the player did not have ball when in fact the player actually still had the ball. This error appears to be mainly attributable to the network detecting shots early, i.e. the neural network classifies the ball leaving the hands an instant before this actually occurs. In cases when the neural network produced an error - the shot was still detected but with an error of 3-4 frames. As the result of this - the ball was released slightly earlier 1/30 sec - 1/40 sec) or slightly later than needed.

CONCLUSION AND FUTURE WORK

We have developed a game platform that uses highly precise motion capture equipment to animate the player and change

the game surroundings if the player's actions impact the game world. Using this platform we have created a basketball game, where a player can dribble and shoot the ball by pantomiming the corresponding moves. In contrast to other existing games, which also employ motion capture (but where the animation of the avatar is done by triggering custom animations which are selected as a result of gesture recognition), the avatar is continuously animated using the actual player movement. The ball behaviour in the game is driven by the physical forces that we read from the player's body and blending them in with the physics system of the game for obtaining believable ball trajectory, bounding and holding behaviour. We evaluated the game with a semi-professional basketball player as well as with a novice player. Both were impressed with the realism of the game and noticed an improvement in their game skills on the real-world court. While in its current form the motion capture equipment is quite expensive for an average consumer, we expect a significant price reduction in the future and hope that advances in nano-technology research will make such motion capture equipment an adequate option for game players. Due to high cost of the necessary equipment, we expect the game in its current form to be mainly used by professional athletes for learning and teaching various game skills. In the future we will investigate using cheaper motion capture alternatives and using 3D displays with portable glasses to address the depth perception problem identified by the evaluator.

REFERENCES

1. XSENS MVN motion capture suit. http://www.xsens.com/en/general/mvn.
2. Berger, K., Ruhl, K., Schroeder, Y., Bruemmer, C., Scholz, A., and Magnor, M. A. Markerless motion capture using multiple color-depth sensors. In *VMV* (2011), 317–324.
3. Bleiweiss, A., Eshar, D., Kutliroff, G., Lerner, A., Oshrat, Y., and Yanai, Y. Enhanced interactive gaming by blending full-body tracking and gesture animation. In *ACM SIGGRAPH ASIA 2010 Sketches*, SA '10, ACM (New York, NY, USA, 2010), 34:1–34:2.
4. Bonnechere, B., Sholukha, V., Jansen, B., Omelina, L., Rooze, M., and Van Sint Jan, S. Determination of repeatability of kinect sensor. *Telemedicine and e-Health 20*, 5 (2014), 451–453.
5. Chai, J., and Hodgins, J. K. Performance animation from low-dimensional control signals. *ACM Trans. Graph. 24*, 3 (July 2005), 686–696.
6. Dutta, T. Evaluation of the kinect sensor for 3-d kinematic measurement in the workplace. *Appl Ergon 43*, 4 (2012), 645–9.
7. Gall, J., Stoll, C., de Aguiar, E., Theobalt, C., Rosenhahn, B., and Seidel, H.-P. Motion capture using joint skeleton tracking and surface estimation. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 0* (2009), 1746–1753.
8. Graves, L., Stratton, G., Ridgers, N. D., and Cable, N. T. Comparison of energy expenditure in adolescents when playing new generation and sedentary computer games: cross sectional study. *British Medical Journal 335*, 7633 (December 2007), 1282–1284.
9. Hairer, G. W. E. *Solving ordinary differential equations II*. Springer, 2010.
10. Ishigaki, S., White, T., Zordan, V. B., and Liu, C. K. Performance-based control interface for character animation. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, ACM (New York, NY, USA, 2009), 61:1–61:8.
11. Jain, S., and Liu, C. K. Interactive synthesis of human-object interaction. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, ACM (New York, NY, USA, 2009), 47–53.
12. Liu, H., Wei, X., Chai, J., Ha, I., and Rhee, T. Realtime human motion control with a small number of inertial sensors. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, ACM (New York, NY, USA, 2011), 133–140.
13. Microsoft. Microsoft Kinnect for XBox 360. http://www.xbox.com/kinect, 2010.
14. Nusman, D. Real-time full-body motion capture in virtual worlds, June 2006.
15. Schlömer, T., Poppinga, B., Henze, N., and Boll, S. Gesture recognition with a wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, TEI '08, ACM (New York, NY, USA, 2008), 11–14.
16. Shiratori, T., and Hodgins, J. K. Accelerometer-based user interfaces for the control of a physically simulated character. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, ACM (New York, NY, USA, 2008), 123:1–123:9.
17. Sony. PlayStation Move motion controller revealed. http://au.playstation.com/ps3/news/articles/detail/item268480/PlayStation-Move-motion-controller-revealed/, 2010.
18. Stern, D., and Hubbard, J. *The Official NBA Encyclopedia. 3rd Editiom*. Double-Day, New York, USA, 2000.
19. Supej, M. 3D measurements of alpine skiing with an inertial sensor motion capture suit and GNSS RTK system. *Journal of Sports Sciences 28*, 7 (2010), 759–769.
20. Tjstheim, I., and Sther-Larsen, H. How to validate a new mr tool? a case study in fmcg. In *ESOMAR-conference proceedings* (Paris, March 2005).
21. Vlasic, D., Baran, I., Matusik, W., and Popović, J. Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, ACM (New York, NY, USA, 2008), 97:1–97:9.