So You Think Your Agent Can Dance?

An Investigation Of Choreographic And Software Structures Within Motion-Capture Driven, Real-Time, Virtual Dance Environments.

Exegesis component for MA by Project

John McCormick

S3087535

Supervisor: John Power

Centre for Animation & Interactive Media

School of Media and Communication

RMIT University, 2010

Contents

Table of Figures

Title

So you think your agent can dance?

An investigation of choreographic and software structures within motion-capture driven, real-time, virtual dance environments.

Abstract

In this research project I have investigated methodologies for creating interactive software environments within the context of a live dance performance. Live dancers wearing motion capture (mocap) technology allowed the streaming of movement data into a virtual, projected software environment. In this project I have looked at both software structures and choreographic structures to see if there are characteristics that might point to appropriate software environments for particular choreographic investigation. The next step was to investigate, by examining the results of various experiments in the nexus of choreography and software design, whether the choreographic structure, not the movement itself, but the underlying organizational architecture, can be used to model the software architecture itself and vice versa.

A research pathway from two dancers using mocap techniques to engage in a shared virtual environment, to a single human dancer interacting with a software agent is described. The research has prompted questions about the possible types and uses of software-driven ‘performance agents’. While it is not within the scope of this research project to develop a fully autonomous collaborative performer agent, a direction for further development is tendered. Existing choreographic structures are compared to appropriate behavior organizational structures in software and in particular game engine

software, in an attempt to elucidate any potential benefits one might have for development of the other. Beginning with simple structures I progress to more demanding choreographic scenarios and propose potential software architectures derived from the requirements of the choreographic structure.

Introduction

When I was first introduced to real-time motion capture possibilities, I was immediately besotted by the often unwieldy and physically encumbering devices and their promise of full-body interaction within a virtual 3D environment. The movement data which was captured by the motion capture systems and sent to the 3D simulation software could effect changes in the virtual environment creating a relationship between the dancer's motion and the resultant visual display from the 3D software. But how close could this relationship become? When I began making work in this genre I tended to think; here is this stream of motion data from the dancer, how can it be visualized in the 3D environment? While, to me, this was a pretty amazing thing to be able to do, the process was largely a one way interaction, the dancer's movement data was sent in to the virtual environment to be rendered in different ways. A form of feedback could be generated by virtue of the dancer witnessing the resultant virtual environment and reacting accordingly, but this was mainly dependant on the sensitivity of the dancer rather than being inherent in the simulation system.

I began to see examples, often in the computer gaming genre, where 3D environments reacted to the Player and often required the Player to respond in kind. I thought that this type of interaction might be extended into the performance works, assisted by motion capture to allow more complex movement potential for the performer than that found in computer games. And so I set out within this research project to find means of allowing the dance and the 3D environment to interact with each-other in more intrinsic ways. The success of alternative interaction models in current gaming environments, such as the eye toy, guitar hero instruments, sing star microphone and in particular the wii controller, has allowed gesture-based interaction in a 3D environment to become relatively commonplace and simple to engage with. As with the gaming genre and many of our daily personal

electronic interactions, it seems reasonable that these electronic elements become more sophisticated in their mode of interaction. There is already a large history of electronic media utilised within dance performance and it seems reasonable to me that these elements should also expand with developments in simulation and interactive techniques. This form of interaction whereby the Player can choose to navigate a simulated environment in a non-linear fashion, and have the environment respond according to its own rules, is an interesting analogy with which to approach the process of creating a performance with live dancer and virtual environment.

I also began to notice that I was working quite separately on the dance and software components of a performance, leading me to question whether that was the only way to work. Given that I was attempting to give the dancer and virtual environment more scope to interact, it seemed strange that my process of development of the two was done quite separately. So as well as considering how to further develop the relationship between the dancer and the virtual environment, I also began considering if there was a way of bringing the two closer together in the conceptual and developmental phases of making a performance.

One way I considered achieving this, was to look at the structures underlying the dance and the software and see if they could somehow be aligned to make it easier to use the same conceptual structures for the development of both. Would it not be nice if I could give the dancer and the software the same instructions and then get them to collaborate on a solution that became the performance?

This research project presents my efforts towards investigating this aim; how to develop a more expressive relationship between the dancer and virtual software environment, possibly using shared structures or goals? To this end I have concentrated discussions of choreography to structures that could possibly be shared by both human dancer and software environment. I have not given an in-depth analysis of the choreography of the live dancer, but rather focused on the intersection of the dancer in the virtual environment and the resultant choreographic effect as a whole. For the purposes

of this research I have concentrated on the potential of the dancer to interact within a simulated environment to create a performance work where the choreography of the virtual elements is as important to me as is the live movement of the dancer. Hence my definition of choreography will include not just the dancer's movement but also the resultant movement of the visual elements of the virtual environment containing the dancer's performance. This focus on structures that allow choreography to emerge in virtual environments was an area I felt required particular attention due to my limited knowledge at the outset of the research project.

In this paper I begin by giving a basic overview of motion capture technology and agent based software concepts in preparation for later sections. I then proceed to introduce the work of other practitioners within the genre of dance who incorporate real-time motion capture as well as some examples of projects that demonstrate motion capture analysis and integration of agent-based software architectures with game engines. Following this I describe the primary hardware and software used in the research, then a description of the research of the project work undertaken with finally a discussion on the findings of the research work and areas for future investigation suggested by analysis of the research projects.

Acknowledgements

I would like to thank all the artists I have worked with in past years, both during and prior to the term of this research. I would especially like to thank Adam Nash for many reasons, Hellen Sky and Ruth Gibson for their choreographic and dance knowledge, Bruno Martelli for innumerable pieces of wit and inspiration, Scott deLahunta who provided numerous, unique opportunities to many artists, myself included, to investigate the role of the dancer in contemporary electronic culture, Paul Bourke who has been an inspiration to me for many years, my supervisor John Power for his support throughout the writing process, Vivek Aiyer, my family and T.

Background

Motion capture (mocap) is the process of measuring and recording the relative position and / or rotations of the limbs of the performer in order to create an accurate representation of the performer's movement. There are many techniques and systems in use to achieve this goal with varying results. I became interested in motion capture as an alternative to other interface options as it afforded the opportunity to use full body motion data within software environments. This allowed different possibilities to the single camera, video based tracking I had used previously. Motion capture allowed a closer representation of the movement in a format readily able to be manipulated on computer, as well as providing accurate position and rotation information with which to interact within the computed environment. A lot of development in motion capture has been driven by the entertainment industry, in particular film and games. This development has led to many possibilities to leverage existing methods from film and games into the performance arena such as the use of a skeletal system to animate software characters in real-time using the motion capture data, the use of physics simulation and collision detection to allow the dancer's data to interact with other elements of the 3D scene, and often relatively easy to use graphic user interfaces (GUIs) with which to construct the environments. The use of a software skeleton which can be manipulated by the motion data is not mandatory but is probably the most commonly used method of transferring the data in to the computer software environment. For representing the performer with a humanoid looking avatar (software representation of the real person) the skeletal system is possibly the easiest method.

In earlier experimental choreographic work, prior to using motion capture, I had been interested in exploring the relationship of performers over electronic networks and the nature of electronic communication. When I started using motion capture I remained interested in these networked

possibilities. Most of the motion capture systems I used transmitted data over a network, usually using UDP (User Datagram Protocol). As the data could be sent to any IP (Internet Protocol) address, it seemed feasible to connect performers in different locations by transmitting their motion data into a single, computer mediated environment, which would usually be running on a centralized server. The successful transmission of the motion capture data is dependent on having access to an open network that allows transmission of UDP streams and has enough bandwidth to transfer the data between the client and host machines. My initial experiments involved the motion capture of two dancers and the representation of their data in a virtual environment. For expediency the dancers were in the same space and the data was transmitted over a local dedicated network.

By Virtual Environment I refer to a software-generated representation of a 3 Dimensional (3D) space in which various objects can act out various relationships. It could be seen as similar to a 3D game, whereby the environment could range from extremely simple to complex, realistic in form to completely abstract. I view the performance projects as existing within a collaborative virtual environment whereby more than one person may interact together within the virtual environment, allowing them to work directly together, or at least to have their data processed together in some manner. This process of collaboration or direct engagement can refer to the relationship between two dancers as in earlier experiments, between live participant and game character as in the case of the game based experiments, or finally between dancer and agent-driven virtual performer (as I will propose in conclusion as scoped for future research).

A linear progression of the research experiments is:

- From rendering virtual humanoid bodies directly to
- Visualising bodies abstractly via kinetic dynamics to
- Visualising the relationship between bodies to
- Creating environment that creates itself from incoming body data to
- Partnership between dancer’s data and software environment to

- Dancer interacting within game environment through a Player character to
- Control structures; finite state machines, branched structures, game character Artificial Intelligence (AI)
- Game Player relationship to Non Player Characters (NPCs)
- Metaphor of Player to Performer and NPC to Agent Performer

In reference to the last point, Agent-based software design has gained momentum in the past few decades as a means of addressing the increasing complexity in modern software development. Software agents are also seen as a means of taking advantage of emerging massive distributed systems such as the internet or subsets of the internet. (Wooldridge 1) Agent-based software has overlapping features in common with other aspects of software design and computing such as artificial intelligence (AI), distributed computing, grid computing and robotics. There are many views of what constitutes an agent, one definition being:

> An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. (Wooldridge 16)

There have been many attempts to describe or define what a software agent is. Many of these descriptions might include notions such as reactivity, autonomy, collaborative behavior, inferential capability, temporal continuity, personality, adaptivity and mobility. (Bradshaw) Agents might exhibit some of these characteristics to a greater or lesser degree. Another approach to the definition of agents suggests that the reason that it is hard to find a consensus on what constitutes an agent is that agency cannot be characterized by a collection of attributes, but rather consist fundamentally as an attribution on the part of some person (Bradshaw 5). This suggests that the key distinction between an agent architecture and another type of smart system is our point of view, or how we regard the system. I think that using both of these approaches to the definition of an agent is useful.

While taking the point of view that agent status can be ascribed according to a person’s views and expectations could lead to almost any software being deemed an agent, conceptualizing software

components as agents and ascribing certain traits to them does change the manner in which they are viewed and I would argue the manner in which they are developed. So while there are some traits that an agent would do well to possess; autonomy, learning, collaborative ability, mobility, the manner in which they are conceived and addressed is also very important. Perhaps it is better to discuss agent *behavior* rather than agent *attributes* as it leaves the definition more open to encompass emerging forms. One means of thinking about agents that I find useful is that:

> ..they locally interact with their task environments, computational or physical, in the course of problem-solving. Responding to different local constraints received from their task environments, the agents can select and exhibit different behavioral patterns. (Liu, Autonomous Agents and Multi-Agent Systems 2)

This suggests that they are able to interact with and respond to their local environments and can exhibit appropriate behavioral responses. In discussing the future of narrative in cyberspace, Murray says of intelligent agents:

> They are improvisers, aware of multiple goals at once and able to change their priorities and behaviors in response to changes in their environment (Murray 227)

While I have not settled on a single definition of agents, mainly as I believe this emerging domain requires a flexible approach to definition, I will focus on these above mentioned attributes: Interaction, Response and Behavior. When later discussing choreographic structure along with software structure, these attributes may give us some concrete parameters around which to compare the two architectures.

Case Studies of Peers

From the outset of this research project, I have been informed by various approaches from other groups to the issues of using real-time motion capture for performances within 3D visual

environments. While some of these examples don't necessarily incorporate motion capture as such, each project reveals various possibilities for the application of technological, design, choreographic and aesthetic principles. The following section is comprised of a number of case studies that highlight other contemporary work in the field.

igloo

www.igloo.org.uk

Igloo is a London based multi-media group whose works contain a strong emphasis on human movement. Igloo was formed by Bruno Martelli and Ruth Gibson. They have used motion capture data and projected environments in many of their works including a web browser based piece, *dotdotdot (2003), Summerbranch (2006)* and *SwanQuake (2008).* The 2004 version of their *Winterspace* installation utilized a video camera based tracking application to cause the wintry projections of the installation to emerge relative to the position of the viewer in the installation space. The effect was to imbue the projected environment with a seemingly rudimentary 'intelligence'. *Summerbranch* and *SwanQuake,* on which I collaborated, made substantial use of motion capture sequences as the basis of the movement behavior of the characters within the game-based environment. They utilized the Unreal Tournament Game Engine.

Figure 1: *Winterspace* Montage, igloo

Bernd Lintermann, Nik Haffner and Thomas McManus

http://www.timelapses.de/

Time Lapses was a collaborative research project exploring '*the relationship between body and media. It focuses on the differences between the body's physical presence in time and space and its mediated visual real-time representations on a screen'* (Lintermann). Time Lapses utilized a Polhemus motion capture system with custom software written by Bernd Lintermann to undertake a series of experimental studies between 2000 and 2003 at the Institute for Visual Media | ZKM Center for Art and Technology in Karlsruhe, Germany. Research was also undertaken at the workshop *Real Time and Networked: Sharing the Body*, hosted by Monaco Dance Forum in December 2002, where I had the pleasure of working with and witnessing the artists' research. The works were documented in great detail and are a pioneering study in the area of dance and real-time motion capture.

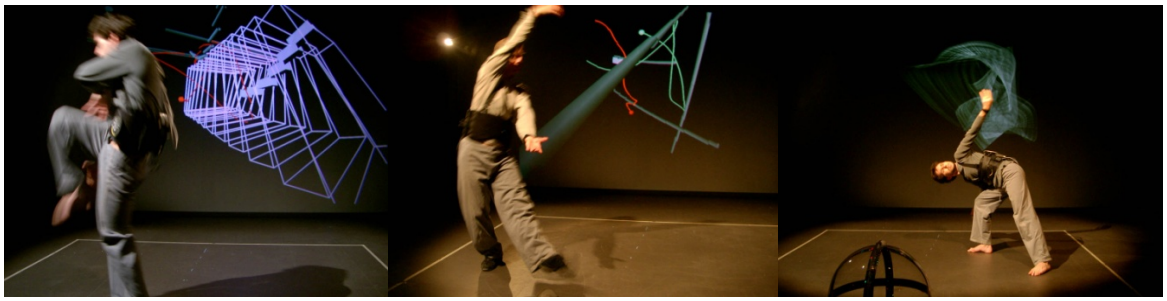

Figure 2: Left to Right, *Accordion*, *Poles*, and *Trace*, Bernd Lintermann, Nik Haffner and Thomas McManus

Troika Ranch

www.troikaranch.org

Troika Ranch was founded by Mark Coniglio and Dawn Stoppiello in 1994. They have been pioneers in the field of interactive dance, creating both innovative, interactive dance works and interactive software (Isadora). Troika Ranch has used a number of self-devised methods for tracking the movement of their dancers and using this information to generate audio-visual environments within their dance performances. *MidiDancer* consisted of a bodysuit outfitted with plastic fibers that measured the flexion and extension of the major joints on the body. When a joint moved, it sent a signal to the computer which allowed manipulation of sound, image and other devices. Troika also uses EyesWeb (www.eyesweb.org) to enable tracking of the dancers and construction in the software of a 2-dimensional skeleton to represent the dancer. The data sent from EyesWeb is then used to control many elements of the Isadora software to produce engaging real-time, projected performance environments.

Figure 3: *16 Revolutions*, Troika Ranch

Freider Weiss

www.freider-weiss.de

Freider Weiss is a Software Artist based in Germany who has collaborated with a number of dance groups and individuals including Chunky Move, Palindrome and Emily Fernandez. His work uses a single camera tracking system to allow the dancer's actions to interact with computer generated environments. While not using traditional 3-dimensional motion capture, Weiss' use of motion tracking to produce rich, interactive visual environments that are responsive to the dancer is an exemplary example of the genre.

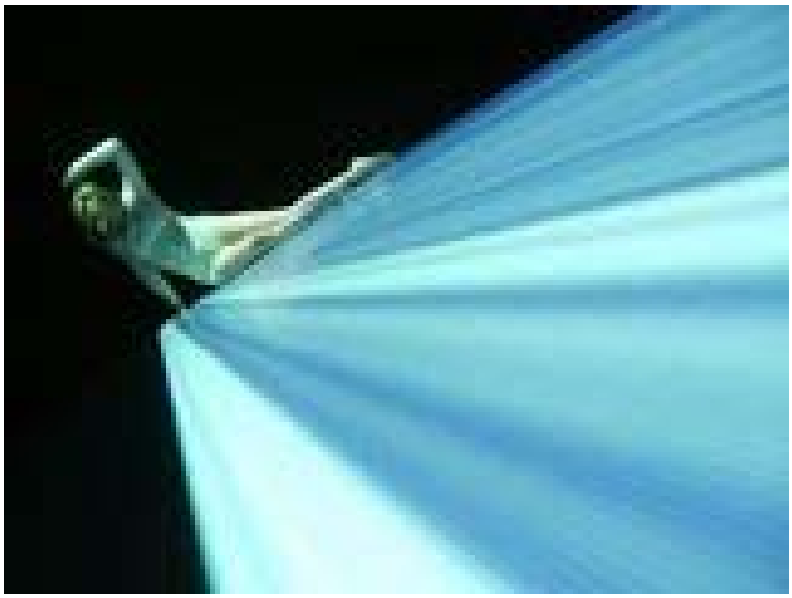

Figure 4: *Glow*, Frieder Weiss /Chunky Move

The OpenEnded Group

www.openendedgroup.com

Paul Kaiser, Shelley Eshkar and Marc Downie (OpenEnded Group)have collaborated with many choreographers, including Merce Cunningham, Bill T Jones and Trisha Brown, to create motion capture driven works. Their work with Trisha Brown *how long does the subject linger on the edge of the volume*…(2005) utilized a motion-capture system consisting of infrared cameras to capture the movements of the four dancers wearing reflective markers. The resulting positional information of the dancers' markers was passed to a software environment written upon the open source project *Field*, authored by the artists. This allowed a "*complex system of analysis and graphic action, an artificial intelligence of sorts where the images' intentionality, memory and tentative grasp of the choreography are enacted.*" (OpenEnded Group) The results were projected in real-time onto a large projection scrim covering the front of the stage, in front of the dancers. This allowed both the projected imagery and the dancers bodies to be seen simultaneously and hence a direct relationship between the dancers and projections to be formulated by the audience. The work is outstanding in the

development of a somewhat autonomous, intelligent system that continually responds to the dancer's actions based on its own set of rules and 'desires'.

Figure 5: ***how long does the subject linger on the edge of the volume..*****, Brown/Openended Group, Image(c) Marc Ginot.**

Dance Education System

The Dance Education System (Leung), undertaken at City University of Hong Kong, utilizes an optical motion capture system to enable students to learn a dance without the presence of a teacher. Animated performances of certain dance sequences are able to be selected and the student, configured within the motion capture system, may perform their version of the dance based on following the animated teacher. The system is able to analyse the performance of the student and compare it to that of the animated teacher, the source of which is motion capture data of an experienced performer. The system can then provide the student with feedback based on the analysis and comparison of the performances represented by the two motion data streams. Some of the interesting aspects of the

system are the emphasis on analysis rather than visualization, and the provision of feedback to the student in order to improve upon the last effort.

RiskMan

RiskMan (Kavakli) takes an agent-based approach to the development of a virtual reality training system. It uses the Unreal Engine, the game engine used in the Unreal Tournament franchise, and which we used in SwanQuake, as the platform for its visual environment and so is of interest with respect to methods of controlling interaction resulting in visual 3D environments incorporating game engine technology. RiskMan is a virtual reality training simulation for risk management aimed at law enforcement officers. What is interesting in its design is the use of separate, interacting, autonomous agents to control different aspects of the simulation environment. It uses five types of agents: a simulation agent decides and maintains the optimal path for training of a trainee; a scripted agent controls negotiation between the trainee and the simulation agent, a human trainee agent is the trainee using the system; the interface agent provides extensible control of keyboard, mouse, gesture and other interface devices; and a communication agent is responsible for speech-based communication with the scripted agent using a natural language generator and a speech synthesizer. The agents interactively negotiate the path of information between the Narrative Engine (which stores and generates the interactive story and scripted agent behavior), the SimMaster (which controls the overall environment based on the current states of the narrative and the trainee), and the Game Engine (where the interacting actors and events are displayed.) The Narrative Engine and the SimMaster are separate programs controlling the decisions of the scripted agents, the Non Player Characters (NPCs) in the game environment. The different components of the system use socket connections to communicate with each other. The current version is a modification (mod) of the Unreal Tournament 2004 game.

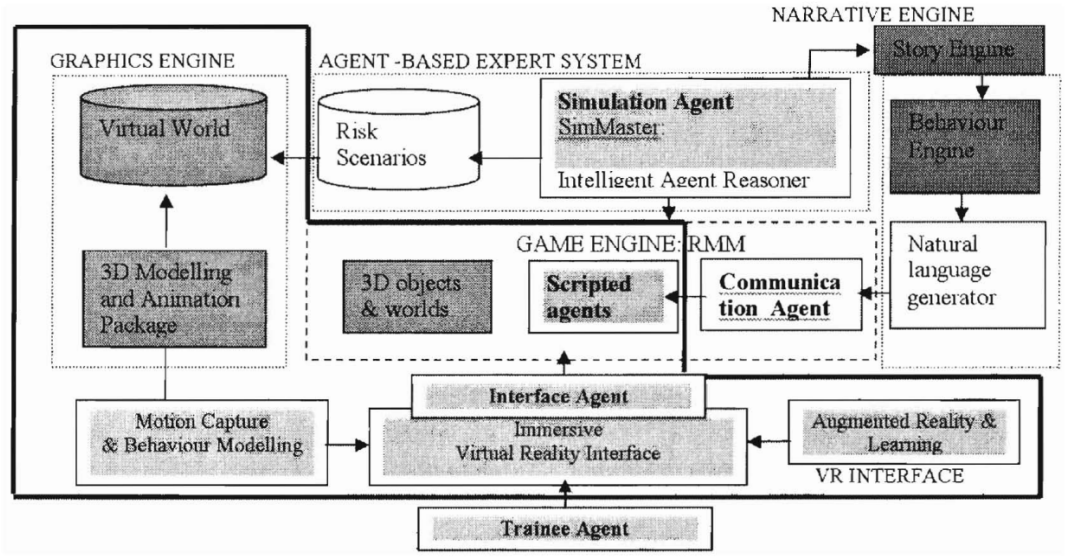

Figure 6: RiskMan Training System Architecture (Reprinted from Kavakli 361)

RiskMan presents some interesting possibilities for the architectural design of complex software projects desiring high levels of interactivity. The splitting of the software into autonomous components allows development of these components independent of each other. This potentially allows many different programming approaches and languages to co-exist in a single work. The use of agents to manage the different components is also interesting as it allows an environment to develop in a persistent manner without continuous intervention from human operators. This agent-based architecture also enables components to be replaced or added without major disruption to other components. For example a different rendering or game engine could be used without changing the other parts of the system.

Design

Description of hardware

The motion capture systems used in this research project all attempt a similar goal, to record the motions of the performer into a format that can be used to represent that motion in digital form. All the mocap systems I used transferred the motion data to a computer over a network where it could be used within a virtual environment. The data was usually used to animate a skeletal representation of the dancer. The raw data as measured from the dancer, however, is not automatically configured in a skeletal format. This process is done in software once the data reaches the computer.

Gypsy mechanical motion capture system

The Gypsy motion capture system is a mechanical exoskeleton comprising potentiometers to measure the rotation of the various body joints and an inertial measurement unit at the hips to measure the rotation of the pelvis. The potentiometers are joined by sliding brass struts that keep the potentiometers in alignment. These sensors actually give fluctuations in electrical voltage equating to relative movement of the sensors, which in turn are converted to digital readings via an analogue to digital (AD) conversion. The resultant numbers can then be used to evaluate the amount of rotational change each joint has undergone since the last measurement. The resultant data is sent via a wireless modem to the host PC where software constructs a skeleton hierarchy which is in turn transmitted over a network to a client PC. The Gypsy has its advantages and disadvantages; as it is a mechanical structure it can be cumbersome and hinders a dancer's contact with the floor (e.g. rolling) and some forms of contact with other dancers. It is a wireless device with a range of approximately 100 metres outdoors and 20 metres indoors, making it quite flexible in terms of range and use in 'hostile' environments, that is, environments in which other mocap technologies would be overly vulnerable to signal interference. Its data is not as accurate as other systems, however the data is not susceptible to

optical occlusion or magnetic interference making it useful for both indoor and outdoor performance. As the entire system is totally wearable, it does not have an external frame of reference like the other systems and as such it is unable to accurately measure absolute position. The global position measurements are calculated from the direction of the performer and the length of each step, which leads to cumulative errors in position. Accurate calibration of the performer helps reduce this error but cannot eliminate it. The advantage of not requiring an external source is that the capture range can be much larger than the other systems and can extend outdoors or through walls.

Polhemus magnetic motion capture system

The polhemus system used was a Polhemus Ultrarack 16 sensor system. The Polhemus system uses a 3 axis magnetic field generator in which the orientation and position of the sensors can be estimated. The device used in the research project was a wired version which caused some constraints on the type of movement achievable without getting tangled in the trailing cables. The system is very susceptible to interference from magnetic structures within the capture volume which can distort the measurements. In a space with little magnetic interference, the results were quite accurate and even with some distortion present, the results were at least predictable. The Polhemus system was accurate enough to allow predictable interaction with in-world actors and could be used in performance situations requiring changes of lighting states and within strong projection fields.

Optitrack optical motion capture system

The Optitrack system used was an 8 camera optical system that tracked 34 optical markers (small reflective balls) attached to the body of the performer. It was the most accurate system used, and the least cumbersome as it had no connecting wires or wearable components aside from the small markers. It did not work optimally with ordinary clothes or costumes as they could occlude the markers' visibility to the cameras. This problem of occlusion was the main problem experienced with the system. If a marker was covered by a part of the body, objects in the environment, or floor, the data could become unpredictable until the marker was again identified by the system. This didn't happen often with the majority of upright movement but floor based movement (e.g. rolling) proved

extremely difficult to track. Another problem experienced was markers detaching from the body during dynamic movements. The system worked best when used with a black tight fitting spandex suit. When used directly within a projected environment the strong beam of light from the projector, as the projection beam traversed the performance space, could interfere with the capacity of the cameras to accurately track the markers. This was most evident when used in a stereo projection set-up, where the viewer wore polarized glasses to provide left and right eye views from two projectors projecting slightly different viewpoints, and it was desirable to have the performer directly in front of the projectors.

Evaluation of different mocap systems in current context

As I have described, the various motion capture systems that I deployed within the research had their own idiosyncrasies which were further compounded by the constraints placed upon them by the particular performance environments employed. The projects were optimally designed for use within a stereo projection environment where the performer was directly in front of the screen so as to give the greatest illusion of the performer being enveloped within the projected environment.

While the optical system delivered the best results in normal circumstances, within the projected environment the cameras suffered from ‘false positives’, a common glitch in image analysis software, where the software algorithms lose track of one or more markers and end up trying to track light sources other than the desired markers.

The magnetic system was not interfered with by the light of the stereo projections and was useable as long as the performance space was relatively free of magnetic interference. The performer had to take care that they didn’t trip or become entangled in the trailing leads. Strategies such as bundling the wires together or hanging them above the floor helped to minimize the risk.

The Gypsy mocap system was the least accurate in that the data tended to drift as there was no mechanism to measure absolute position. This only tended to be a problem where it was desirable for the dancer’s avatar to interact directly with the virtual environment in a way that maintained a

meaningful registration with what was happening in the real world space, and accurate positioning of the body was necessary. The internal rotation measurement of the joints tended to be reasonably accurate and sufficient for most purposes.

Description of Software

Rather than try to construct a performance software environment from scratch, I chose to see what software existed that could either fulfill my requirements or provide a foundational basis for development.

Requirements included:

- The ability to interface to motion capture devices and manipulate the data streams
- Passive stereo projection capability
- A means of extending the software through a scripting interface or ideally a plug-in architecture
- Relative ease and speed of development given sole developer

I was interested in software platforms that took care of as much of the commonly used components as possible leaving me more time to concentrate on items specific to my research projects such as motion capture interfaces, analysis and manipulation of data and visual representation of the data.

Alias Motionbuilder

Motionbuilder is primarily developed for capturing, editing and visualizing motion capture data. It is useful as a rapid prototyping platform for the type of work undertaken in this project. It contains a basic building-block style scripting interface that is fast to work with but can be limited in functionality. There is a lack of many of the functions found in modern visualization and game engines such as collision detection, Artificial Intelligence (AI) scripting and interaction interfaces.

Developing plug-ins is not as easy as in other visualisation environments. It is available for Windows and Macintosh platforms.

Act3d Quest3d

Quest3D is a DirectX based 3D development platform useful for developing games, architectural walk-throughs, simulations and scientific visualizations. It is available for the Windows platform only. It has a comprehensive framework for developing plug-ins with which to extend the development environment and with which I wrote plug-ins for the three motion capture systems used in the research. Quest3D is not just a game development environment but is suitable for a range of simulation environments.

Unreal Engine

The Unreal Engine is the underlying technology used in the Unreal Tournament series of games developed by Epic Games. It comes with a comprehensive editor, UnrealEd, and scripting language, UnrealScript, making it a favourite of the game modding (modification) community. The online community[1] and support structures make it very useable for 3D visual projects. One problem can be that the underlying platform, the Unreal Engine, is optimized for first person shooter (FPS) games and can take a fair bit of effort to create other types of structures within the software. It does not come with source code unless extremely hefty license fees are paid and hence it is not easy to extend the game development environment with features such as motion capture capability.

Torque Game Engine

The Torque Game Engine (TGE), developed by Garage Games, is the engine that was used on the Tribes franchise of online capable games, originally developed by Dynamix, a subsidiary of Sierra Entertainment. The content of the games was stripped out by Garage Games and the actual game engine offered to game developers, mainly independent developers. It is one of the few engines that

[1] See, for example: http://wiki.beyondunreal.com/ (last accessed 24th Mar 2009).

offer the full engine source code with all licenses. It is also quite affordable for a commercial game engine.

Unity

Unity is a multi-platform game development tool developed by Unity Technologies, capable of publishing for MAC, Windows, Wii and iPhone. It comes in Indie and Pro versions. The Pro version is required to add plug-ins in order to extend the basic development environment.

DANCE

The Dynamic ANimation and Control Environment (DANCE), authored by Ari Shapiro and based on earlier work by Victor Ng-Thow-Hing and Petros Faloutos, is a software package for physics-based character animation and simulation. It is a self contained program that offers a variety of methods for generating character animation by simulating how the character would behave under prescribed physical conditions and by incorporating motion capture data. It can be compiled to run under Linux, Windows and OSX. It would be feasible to take methods employed in DANCE and incorporate them into your game engine of choice, providing the engine can be extended through a plug-in architecture or similar method. DANCE itself can be extended with user created plug-ins. One of the aims of DANCE is to offer a platform in which character control methods can be tested and evaluated.

Description of projects

Overview

The performance–based projects undertaken were conducted with a view to being short performance pieces set in a proscenium style setting. The audience would be situated at one end of the space with the performer at the other. A large screen would fill much of the wall behind the performer onto which a stereo image is projected. The performer is in front of the screen to allow the audience a

view of both the performer and the resultant real-time interpretation of the motion data and to emphasise the visual effect of the stereo projections in surrounding the performer. The audience wears glasses with polarized lenses in order to perceive the stereo effect. The works were conceived as short 3-5 minute pieces to allow rest for the audience from having to cognitively synthesise the stereo fields and this was seen as a suitable time in which to experiment with the limited parameters chosen for each study. The earliest works experimented with two performers in different motion capture systems whose data was streamed into a single visualized environment. The performers could be in the same location or in remote locations.

While the above was an ideal situation, the experiments were not all conducted under these conditions. Some works (2 Avatars, Nova) were performed and recorded in a studio with single screen, non-stereo projection, or with a monitor for feedback. Tosser and Lines were presented in stereo but the performance space was rather limited in size, making calibration of the systems challenging. The motion data from all these performances was recorded so that the simulation could be reproduced, often with changed parameters in order to see the effect of changed interactions. The software could not discriminate between the live motion capture stream and the recorded motion capture stream, which allowed testing of the software environments without the requirement of a full performance setup including dancers on call. As the notion of shared structures between the live motion capture performer and the virtual environment became more paramount, I focused on the area where the two intersected, for me this was the software environment where the dancer’s data could transform the environment and/or the environment could respond to or expand upon the incoming motion data. This is reflected in the photos and documentation which predominantly represent the shared virtual environment as the dancers’ motion data enters it. The use of recorded data made experimentation with the software environments much more practical.

2 Avatars

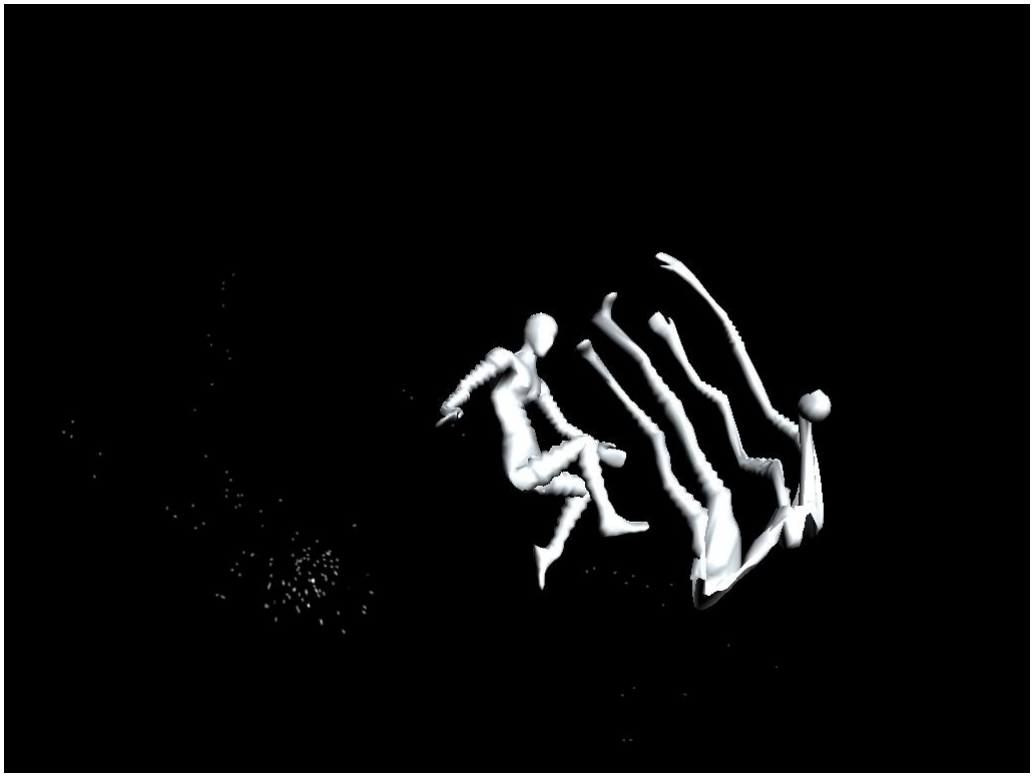

Figure 7: ***2 Avatars***

2 Avatars (Figure 7) was a live performance within a virtual environment that investigated the use of non-rigid humanoid representations as avatars for the live dancers. The first sketch of *2 Avatars* was created during the *Motion Capture Tech Laboratory "Real Time and Networked: Sharing the Body"* hosted by Monaco Dance Forum in December 2002, and moderated by Scott deLaHunta (deLaHunta). In previous work, such as *CO3* (2002, Company In Space, Manchester Commonwealth Games) the avatars used were deformable solid meshes of the type seen in many 3D games and animation. The direct mapping of the motion data to these humanoid meshes tended to look rather mechanical in that simply rendering the sampled motion tended to look pedestrian compared to the multi-layered sensory engagement one can experience when witnessing a live performer. The use of soft-bodied avatars was an attempt to give the rendered motion unique attributes derived from but not

necessarily mimicking the live performer. The virtual bodies had an amount of elasticity that was exaggerated as a function of the velocity and direction of travel of the avatars. This methodology allowed for the visualization of the raw motion data along with some of the inherent dynamics of the movement.

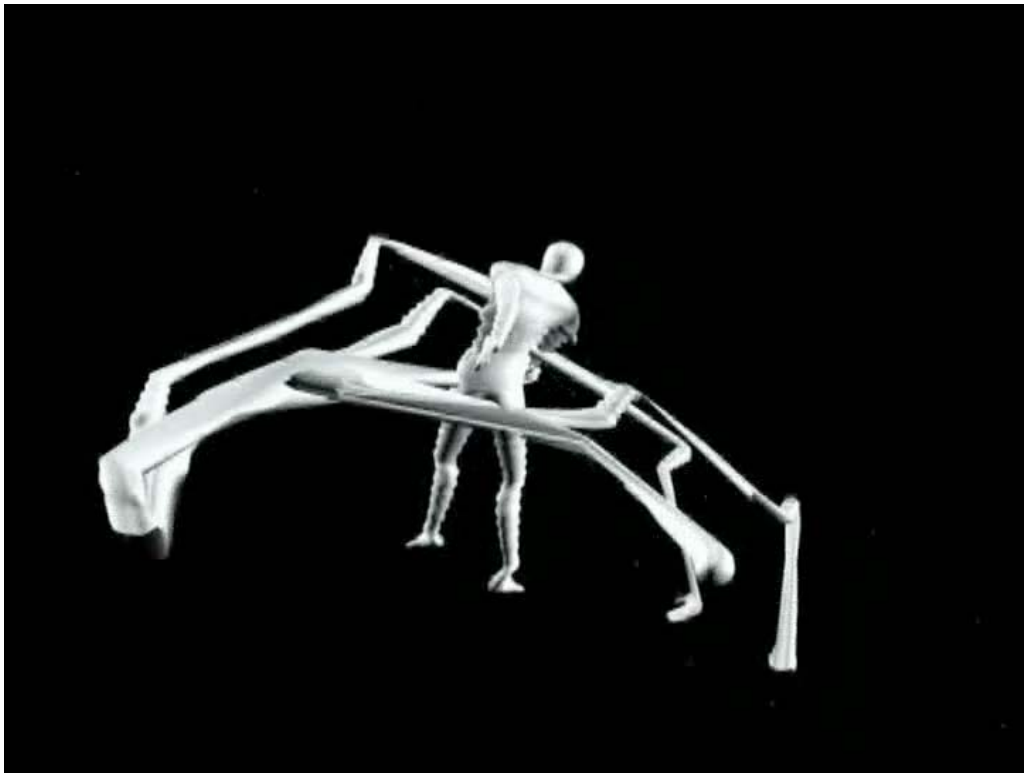

Figure 8*: 2 Avatars*

Another aspect investigated in *2 Avatars* was the use of one motion dataset to influence another (Figure 8). The arm gestures of one dancer were used to sweep the second avatar in arcs around the so-called 'world centre' of the 3D Cartesian axes. This exaggerated the elasticity of the target avatar and created a unique relationship between the dancers within the virtual representation. *2 Avatars* used a simple mathematical transformation to achieve the stretched effect; taking the speed of the hand movement from one performer and using it to apply an amount of drag on the second performer's avatar. The crossing over and redirection of data from one stream to another proved to be a simple but effective mechanism for extending the visual possibilities of the projected environment.

Tosser

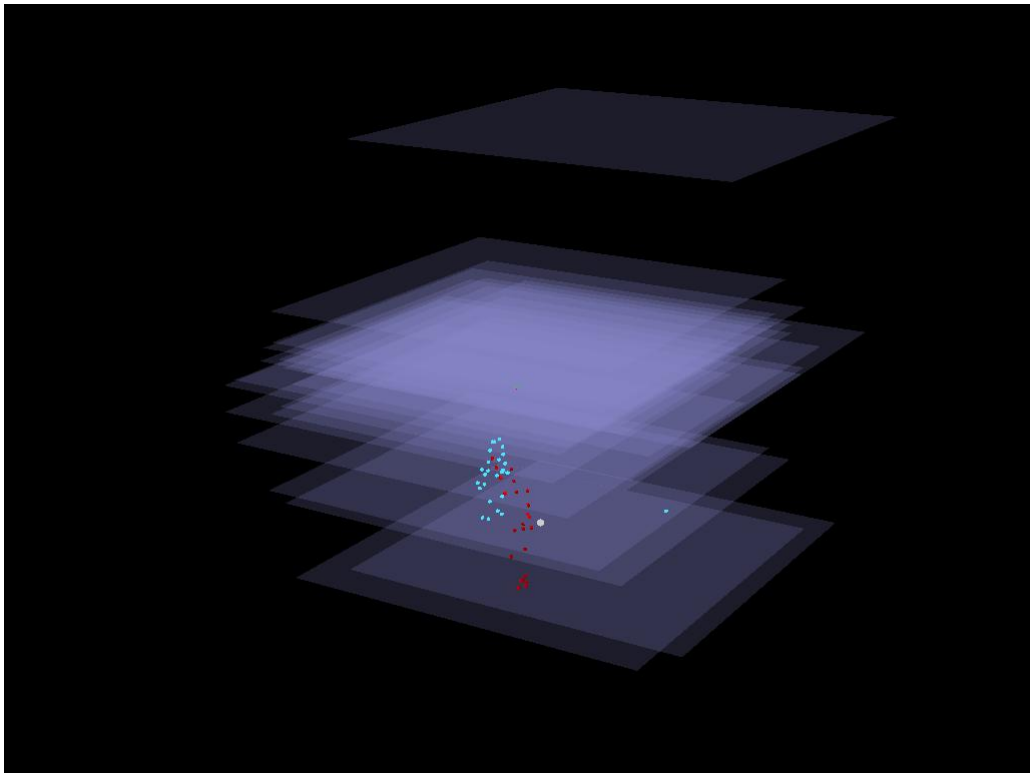

Figure 9: *Tosser* showing dancer joint nodes

The results of *2 Avatars* prompted me to look further at visualising the dynamics of the live motion datasets and creating further recombinations of the two datasets of two live dancers. The treatment of the motion streams as datasets rather than simply representing movement allowed for the conception of many possibilities for visualisations based on the live data. The method of combining the datasets was taken further in *Tosser* (Figure 9). *Tosser*'s first incarnation was in *Digidance 1*, a dance and technology workshop hosted by Essexdance in Chelmsford UK, in February 2004, and moderated by Scott deLaHunta. In an attempt to treat the combined dataset as a single entity and to focus on their dynamic properties, the humanoid bodies were dispensed with in favour of more geometric shapes. This was in order to concentrate on the inherent patterns and dynamics of the

combined data. A stack of virtual glass sheets were individually propelled around the virtual space according to the combined velocity of the joints of the two dancers. So one sheet might respond to the combined velocity of the dancers' right wrists, the greater the velocity the further the sheet was expelled away from the centre of the volume. The sheets would return to their original starting points if no impetus was applied by their corresponding body parts. The result was a geometric pattern that could be seen constantly reconfiguring itself according to the summation of the motion datasets as derived from the dancer's physical exertion (Figure 10).

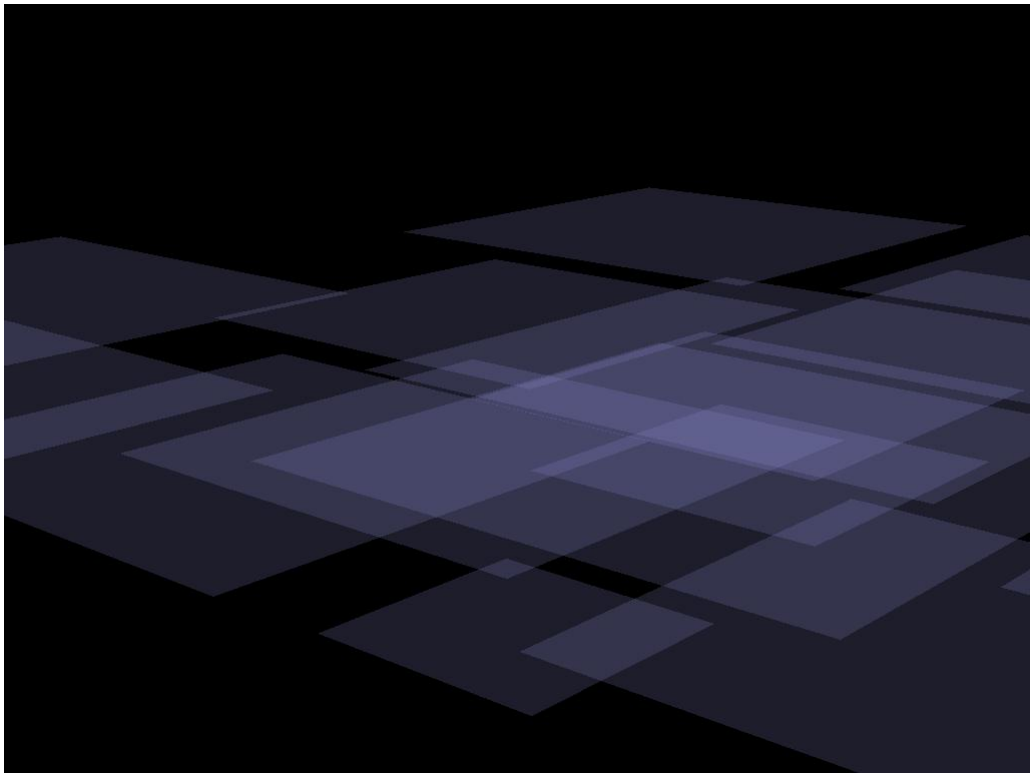

Figure 10: ***Tosser***

Tosser began a line of research in to the creation of virtual environments external to the dancers' bodies. Instead of imagining the dancer's movement being represented by a humanoid avatar or even an abstract visual environment, the virtual environment was imagined as an independent entity that could receive the incoming streams of motion data and employ it to visualise the results. This led to

setting up filters for the incoming data that could be applied to objects in the virtual environment rather than the data being directed onto an avatar skeleton directly.

Lines

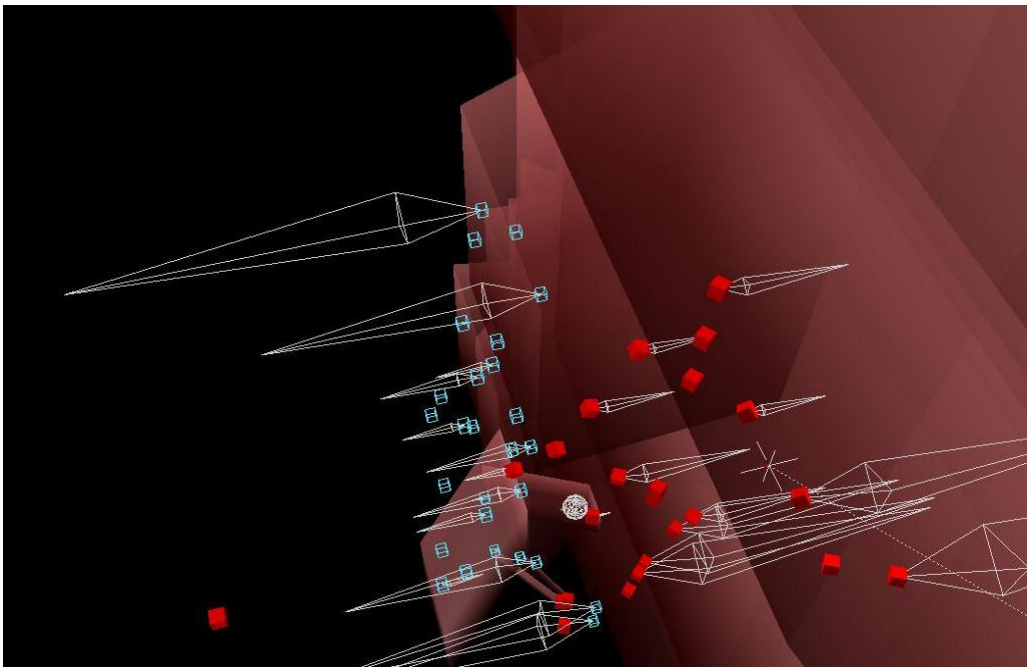

Figure 11: *Lines* showing dancer skeletal nodes (Dancer A-blue, Dancer B–red)

Lines (Figure 11), was also conceived at Digidance1 in Chelmsford, and along with *Tosser* and other works, was first performed at the Swinburne University Virtual Reality Theatre, Department of Astrophysics and Supercomputing, in 2005. *Lines* further explored the method of filtering the datasets through 'logic blocks' before visualising the results. As in *Tosser*, the visual environment was conceived as being derived from, yet independent of the original movement data. In *Lines* the dancer's motion data was sent as UDP streams to the computer running the visualisation. The dancers could be in the same space or in distantly remote spaces. The work created a visual entity between the dancers. The joints of the bodies were joined by cubic objects that illustrated the combined speed and origins of the respective joints (Figure 12).

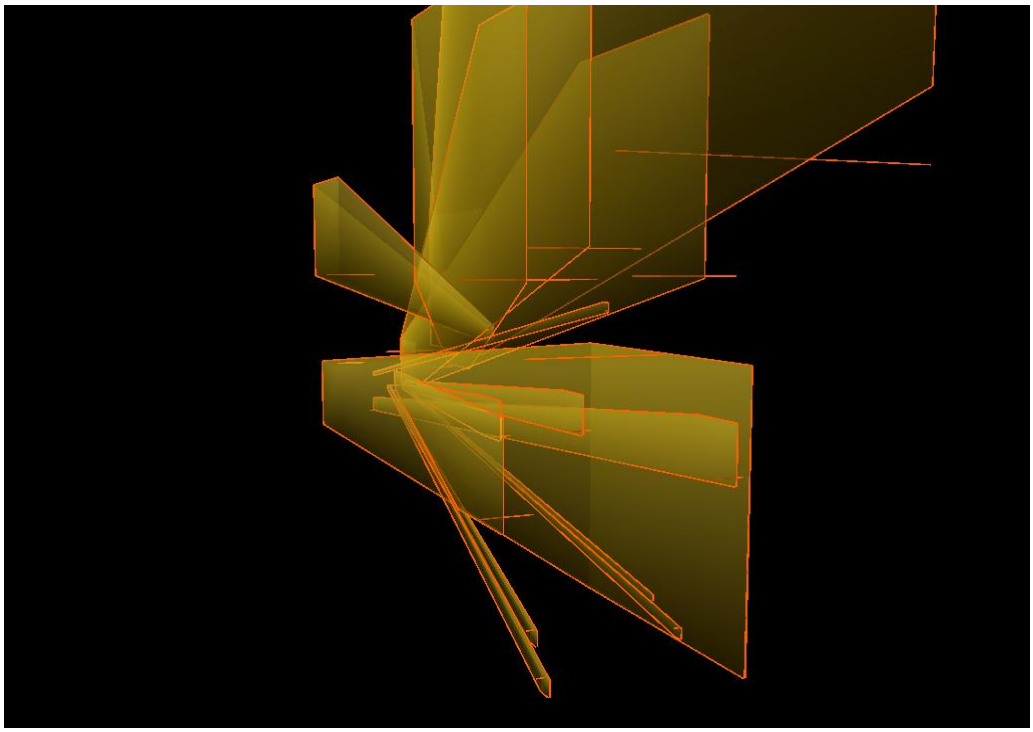

Figure 12*: Lines*

The resultant geometric volumes grow and contract in response to the dancers' combined movement dynamics. Lines was conceived as a collaborative networked artwork where the dancers could be in any geographical location as long as they had access to suitable network infrastructure to carry the motion data to and from their partner's location.

Nova

Figure 13: ***Nova***

Nova (2004) was presented at the International Symposium on Art and Technology, Utah, USA in September 2004 with Company In Space. *Nova* was a small investigation into the use of behavioural 'states' to control the behaviour of particle generators. The particles had two states, dormant and attached. In the dormant state the filter was represented by a particle-system generated fire that emanated from a fixed position. In the attached state the particles split off and attached themselves to parts of the dancer's avatar. The behavioural states could present the particles in a more traditional mode, that of a fire as found in many animations or games and having a fixed location, or change to appropriate the motion dynamics of the dancer in portraying the trajectories of the particles. The states could be triggered by user intervention (keyboard), midi control data or from triggers from the dancer's movement. This represented a slight departure from earlier works in that the method of filtering the motion data could be changed within the current software environment without loading in a new scene. While control of the states was governed by direct intervention by the artist, it also led me to considering the possibility of the software or external agents making appropriate decisions on the current appropriate state.

SwinGEE

In *SwinGEE* 2004-2005, (Figure 14) a research project conducted with students at Swinburne University[2], a Player character was driven by the Gypsy mocap system allowing the Live Player/Performer to interact in a virtual world with a Non Player Character (NPC). The Torque game engine (www.garagegames.com) was used and was augmented to receive mocap data from a live 'Player' wearing the Gypsy and to project the virtual environment in passive stereo. The motion data was not used to drive the Player skeleton directly, through applying the rotation data from the gypsy to the Player skeleton, but rather the gypsy data was used to proportionately blend pre-recorded locomotion animations available to the Player character in order to effect movement. For example the arms of the character had animations for a range of rotation on the X, Y and Z axes for each of the arm segments (shoulder, elbow and wrist). These animations were blended together according to the rotation angles of the live player's joints, sent from the gypsy. This method, of blending together different animations to create alternative versions of character movement is often used to animate NPCs and 'normal' Player characters and was the easiest method to implement in the Torque engine at the time. While we could use the live mocap data to drive the animation of the player character and calculate an approximate collision response so that the Player could physically interact with the NPC in the virtual environment, including being able to use a sword and other weapons against the NPC, the responses from the NPC were very rudimentary in scope, relying on the pre-recorded animations available to it.

[2] Conducted at Swinburne University Department of Supercomputing. Team members were Jackie Ng (team leader), Magnus Nordby, Sokra Cheng, Chung Ming Sin, Ricky Soegandy, Ali Rajani and Ting Khen Hui under the supervision of Paul Bourke and John McCormick.

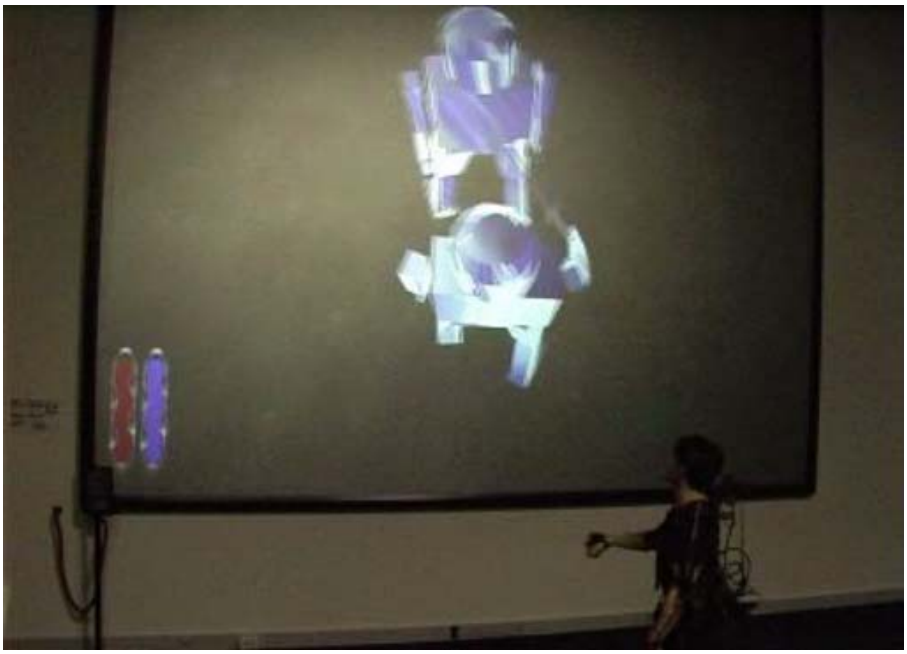

Figure 14: SwinGEE: Mocap player vs NPC in Torque game engine

One problem with the project's implementation in Torque was the calibration of Player locomotion in the game world. If movement in game space was determined by the amount of distance covered as the live Player walked, s/he would cover relatively little area within the game world, certainly not enough to find and interact with the NPC's. This was compounded by the limited physical space available to the live Player in front of the screen in the Virtual Reality Theatre in which the project took place. To enable the Player to traverse greater distances the leg locomotion used regular walk, run and jump animations typical of those found in most games, which were controlled by the keyboard and mouse. These were blended in with the live motion capture controlled animations of the upper body. Thus the player character was controlled by two people at the same time, one controlling upper body movement using the Gypsy mocap system, the other using the keyboard and mouse to control ambulation within the game environment. It was possible to also control the leg motion with the gypsy, and given a larger space, the live Player may have had enough room to traverse more of the game world with the Player character mimicking the live Player's leg movement. However we desired for the Player to be able to traverse quite large game environments and the blending of traditional controls for the legs and motion capture control of the upper body

allowed this. Methods to give the live Player control of the locomotion process, rather than delegating it to a second person on keyboard and mouse, were considered including the use of a weight transfer board that could be attached to a joystick mechanism to allow the live Player to lean in the direction they wished to travel and thus control the ambulatory animations.

Another issue encountered in the SwinGEE research was the manner in which contact between the Player character, NPC's and the game environment was detected. Torque 1.3 used a collision mesh system to detect contact between game elements. The collision mesh is an invisible geometric object that covers some or all of the area of the game object, and it is with this collision object that the game engine performs collision detection in order to ascertain certain physical interactions. Many engines at the time used quite primitive collision hulls, even for the Player and NPC characters, often a simple vertical cylinder that covered most of the character's body. This meant that contact between Player character and NPC was based on the primitive shape of the collision cylinder rather than the shape of the character's body. One solution was to create individual collision objects that closely fitted the basic parts of the character's body shape. This tended to slow down performance a little as the collision calculations became much more complex. A future version of Torque Game Engine and many contemporary game engines use 'Poly Soup' collision or 'Mesh Collision' which allows the actual character mesh to be used to calculate contact with other characters and the game environment. A more sophisticated form of collision detection than the one we managed in the SwinGEE research would be desirable if complex physical interactions between the Player character and the NPC were to be simulated.

While SwinGEE was successful in incorporating mocap data to allow a synthetic character representing the live Player to interact within a game environment, the implementation was not trivial in that the source code was rather convoluted and required very strong programming skills. The amount of programming required to develop game levels would be enough to preclude development of environments by most sole developers. Subsequent to the SwinGEE project I researched a number of other game engines as alternatives to Torque including Quest3D and Unity. After writing plug-ins

for the motion capture systems for use in Quest3D, I found the character animation framework in Quest3D to be rather difficult to use. While the process of receiving and managing the mocap data in Quest3D worked well, transferring the data on to the skeletal structure of 3D characters in order to animate and interact with them gave unpredictable results and was at times quite frustrating. Using the motion data to animate or control other objects in the Quest3d environment worked well, but the under-developed structure of its skeletal animation system prompted me to look further afield for an acceptable solution to this particular part of my research. Currently I am working with Unity whose character animation framework appears to me to be far more useable than Quest3D or Torque for the purposes of using motion data to allow a live performer to animate a synthetic character and interact with other digital characters within the game environment.

SwanQuake, SummerBranch, Unreal Engine

I cannot remember exactly when the concept for SwanQuake arose, but I remember it was in the igloo studio one evening and Ruth Gibson and Bruno Martelli (igloo), Scott delaHunta and myself were keeping warm with a few heat-inducing beverages (I'd say it was London winter, probably after the Digidance workshops in February 2004), when the title popped up. The title said it all, dance (as alluded to by the well-known classical ballet Swan Lake) and game engine (as exemplified by the Quake engine and game, being one of the first 3D first-person shooter games). Bruno and Scott were willing to engage in some collaborative game engine research, so we started investigating the then current crop of game engines with a view to finding a platform for *SwanQuake*.

We considered a few engines, most in the First Person Shooter (FPS) genre, as these came with editing tools and some with exporters for 3D creation tools which enabled importing motion capture sequences into the engines. We looked at (and thoroughly tested) the engines behind Medal of Honor, Call of Duty and others before eventually settling on the engine used for Unreal Tournament 2004. From this investigation of employing game engine technology to house dance content, *Summerbranch* was premiered at Artsway in 2006 and *Swanquake* at V22 in London in 2008. Both *SwanQuake*

(Figure 15) and *Summerbranch* used relatively long motion capture sequences for the motion of the characters. The works were usually projected onto gallery walls as installation works where participants took the role of the Player and navigated through the game worlds interacting with the environments and the Non Player Characters (NPCs). The Player interacted within the game using mouse and keyboard and interactions were largely determined by pre-defined scripts. Interactions between the Player and NPCs were based on proximity and contact (through collision detection) of the Player with the NPCs. Responses by the NPCs included disappearing and changing the current movement phrase. Motion capture sequences were used to animate the Player and NPC, and their sequencing was controlled by scripts. *SwanQuake* and *Summerbranch* both used slightly modified versions of the standard navigation and interaction techniques used in the Unreal engine. While the interactions within the works were not unusual for this type of game engine, they suggested a number of interesting ways to perceive the relationship between a live protagonist (Player) and digital collaborator (NPC).

While working on *SwanQuake* I investigated the structures used within the Unreal game engine to organise the movement behaviour of NPCs and how they react to player characters. In the Unreal engine the NPC is often called a bot, (probably a derivative of robot or software robot), whose simple behaviour can be controlled by simple functions that are 'called' at every frame[3] or in response to some external stimulus e.g. being bumped by a player character. For behaviour with a limited palette, calling functions to define responses was often adequate to enable basic reactivity in the NPCs. In order to give a more complex range of responses and to simulate intelligence for the NPC, states are used. These states can define a broader range of actions for the NPC in response to changes in the game environment. In *SwanQuake* specific responses were programmed depending on the relationship between the Player and the NPC which could change over time or as events dictated. Some simple responses were encoded; for example, the NPC would become transparent, less visible, as the Player

[3] The frame rate can vary when the Unreal Engine is running, from anywhere between once to over one hundred times a second, although 'real-time' tends to presuppose at least 25 frames per second.

approached, creating a sense of elusiveness. (Figure 16) Choreographed actions were performed when the Player bumped into or touched the NPC or when the Player came within a certain proximity to the NPC. Though there were limitations to the types of interactions that were possible in the Unreal Engine, partly due to its fundamental architecture as a First-Person-Shooter (FPS) style game engine, it did prompt me to look further at using Artificial Intelligence (AI) as a mediator between the motion activities of a live Player/Performer and a digital Partner/NPC.

Figure 15: Swan Quake, igloo

In many of the previous experiments two dancers were performing together in an increasingly mediated environment. The work with game engines encouraged me to view the software entity (a.k.a. bot, NPC, agent) as a potential partner that could mediate the relationship between two networked dancers, or could replace the remote dancer and potentially become the dancing partner, responding to movement stimulus from the live dancer's motion data.

There are a number of strategies used in the Unreal engine to govern interaction with bots and other in-game elements: scripted sequences, states, AI, interactions based on the relationship with the player and other bots and interactions based on triggers, zones or states embedded in the virtual environment itself. The choreographic potential inherent in game engines is what is employed to also create machinima artworks.

Now the choreographer faces a challenge similar to that of a games developer. Given a set of motions, how do you build an architecture for a larger ensemble of motions? How do you string them together? Starting from one motion, you could consider any number of possible movements to join it to. (Kaiser 6)

Considering the game engine not as a closed environment but as a space for collaboration between a live dancer (akin to the Player) and a digital partner (akin to a NPC) yields some interesting potential for exploring the duet form in a synthetic environment.

Figure 16: Swan quake, igloo

Discussion

Path of Development

While, through my projects, I began investigating the use of motion capture to allow two dancers to engage in a shared 3D synthetic environment, I have moved more recently towards developing automated mechanisms that enable a dancer to engage with a synthetic partner. This partner could be of humanoid or abstract representation or take many visual forms. The choices of representation are specific to the work being undertaken and can be viewed as separate to the structures that govern their behavior. I am also going to limit the discussion here to game engine and related software environments as that is the path I have taken, and am currently taking for present and future work. Also, to narrow the context of this discussion to a manageable level, I will imply the synthetic partner to be a valid substitute for a human partner in terms of experimenting with movement responses and capability. I haven't intended to necessarily mimic complex human movement and responses in software, however I am interested in interaction with a character that is plausibly recognisable as being of human derivation. This does not mean the character would have to pass a visual equivalent of the Turing test [4], whereby an external viewer would have to ascertain which is the real human (data), and which is the synthetic. The human movement basis is partly dictated by the use of a humanoid skeleton for both Player and NPC, and I have emphasised physical movement rather than emotional or intent driven behavior. This is still a wide field of play as, in *Tosser* for example, while the pattern of behavior of the geometry is based on real human motion, i.e. movements that are constrained by the dancer's biomechanical structure, the result does not necessarily look like bipedal motion. While the

[4] In 1950 Alan Turing posed an experiment to determine a machine's intelligence. If a third person in carrying out a text based conversation with a computer and a person could not tell which was the person and which the computer, the computer was deemed to be able to 'think'.

manner in which the synthetic partner is represented will vary the degree to which human traits might be attributed it, all of the projects I have undertaken have used a skeletal representation of the dancer or character to visualise the movement in the 3D environment. This use of a skeleton to represent the human body is a convention prevalent in most of the software used in this project. The motion capture systems can all make use of a skeletal representation on which to model the movement of the performer. This skeletal paradigm is expedient in that it allows different systems to exchange motion data in a relatively consistent and predictable manner.

Use of a Skeletal Hierarchy to Animate Synthetic Characters

The skeletal hierarchy becomes an important tool for capturing, transferring and representing the dancer's motion and also as a means of representing motion in a synthetic character. It is through these skeletal hierarchies that most of the interaction between dancer and synthetic character has occurred. In earlier works, such as *2 Avatars, Tosser* and *Lines*, the two dancers were also represented through a skeletal system and most of the calculations of interaction in the shared environment and resultant responses were performed on parts or the whole of the skeletons. Characteristics of the movement gleaned from the motion data included proximity of one dancer to another, and to other parts of their own bodies, speed of body parts, direction of facing and movement, relative and absolute position in space, angle of rotation of specific joints and of multiple joints, the latter which can be used to predict the occurrence of specific gestures. Using skeletal models as a basis on which to analyse and formulate responses to incoming motion data from two dancers was very useful in that it provided a common framework for using different mocap systems in the same work, allowed a fairly easy transfer of resources from one software environment to another, gave useful feedback to the dancers as to what effect their movement was having within the virtual environment and allowed the viewer to more easily interpret the movement relationships between the live dancer and their virtual representation. As a consequence I continue to pursue this model in looking at relationships between a live performer and a synthetic character.

In the film and game industries the most common way of animating characters is to create a skeleton that is then used to deform and hence move the mesh or skin of the character. This method of animating characters is found in most 3D creation software packages and is also supported in nearly all game engines. It is also the method employed to visualize real-time motion in software packages such as MotionBuilder (see above).

How can this ubiquitous skeletal arrangement be employed in the context of a live performance between dancer and agent? A similar if not identical skeletal structure can be used for both recorded and live motion data. This presents the possibility for editing and recombining the motion from human and synthetic characters both individually and together. In the Unreal Engine, Quest3D and Unity the default method of animating characters is using skeletal animation. Motion data can be imported into a 3D animation package and transferred to a character via the skeletal hierarchy. The animated character is then exported to the desired game engine and the animations can be used to represent Non Player Characters in the game environment. There are a number of methods used to control these available animations, They can be played in sequence, looped, blended to form new animations, parts of one animation can be overlaid upon another animation e.g. an animation of waving arms played over a running animation, or they can be combined with an Inverse Kinematic [5] system which alters the original animation to allow fine tuning of things like grasping objects or maintaining foot contact on uneven surfaces. The NPC movement can also be treated the same as a live character, by sending a motion data stream to animate the character. This is a typical method used to visualize the motion data from a live dancer. The live motion data stream is connected to the skeleton of a character to animate it in concert with the dancer. A motion stream can be simulated, either composited from existing motion sets or generated in software, that looks to the character like live human motion data. It follows then that this would allow for the generation of motion responses

[5] Inverse kinematics is a method of controlling flexible joined objects, such as the bones in a 3D skeleton, so they can be manipulated to achieve required actions. For example, the feet might be moved up or down to adjust for uneven terrain and the rest of the skeleton (shin, thigh, hips, and onward up the chain) would be adjusted appropriately to compensate. Often the adjustments try to maintain human biomechanical limits.

for the synthetic character based on many methods; modification of the live dancer's motion, re-compositing of recorded motion sets, pure software or mathematical generation of motion sets or any combination of these. It would also allow the logic that determines the responses to be either within the game engine environment or outside it, sending the resultant motion stream into the engine as you would a live stream.

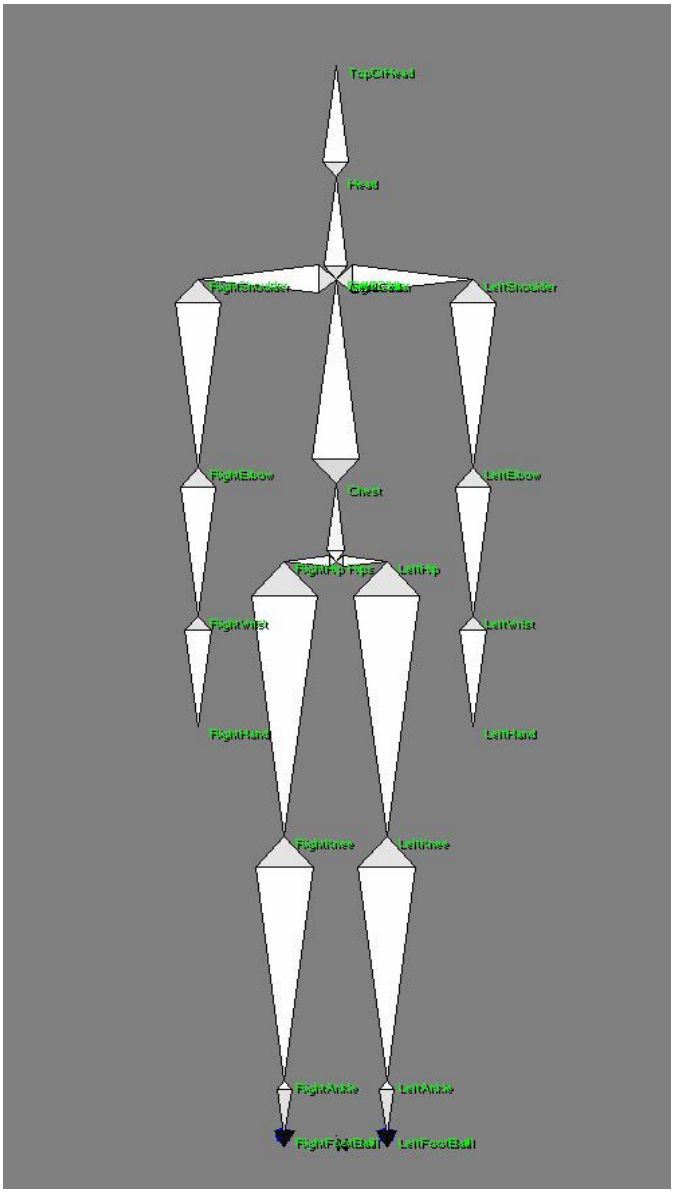

Figure 17: Typical Skeletal Heirarchy

Behaviour control structures in the Unreal Engine

During the course of working on *Swan Quake* I investigated many of the Non Player Character (NPC) control structures available within the Unreal engine. The scripted sequence found in the Unreal Engine, which has a counterpart in most game engines, allows the developer to create more complex behavior from basic components. In movement terms, a NPC might have a number of short animations within its repertoire that could be put together to form a longer sequence of movement. This ‘sequencing’ type of choreographed behavior could be seen to approximate a traditional choreographic form where the dance is composed of known movement parts that are performed in a linear sequence from start to finish and also reflects certain digital methods of composition in music software and, more recently, in animation and motion graphics software that use repeated ‘tiles’ of motion or time-based motifs to run in preordained sequences or to run when triggered by certain other events. An example might be a ballet sequence where the dancer performs known, predetermined steps in a predetermined order. This is a common choreographic device used across many dance genres and may even be said to be typical of much western contemporary and classical dance. From the available arsenal of character control methods present in the Unreal game engine, the scripted sequence is a suitable structure for a synthetic partner to accompany a dancer undertaking this form of choreography. The scripted sequence was useful for causing the NPC to undertake a set sequence of movement animations in response to an event such as interaction with the Player.

Figure 18: Unreal characters controlled by a scripted sequence

While using the Unreal Engine in *SwanQuake,* as a container for recorded dance activity, I started to notice some similarities between the control structures built in to the engine and some choreographic structures I had previously read about. In particular I found strong resemblances between the structure of a scripted sequence as used in the Unreal Engine and Steve Paxton's *Satisfyin' Lover*. (

Figure 20) From his early days as a choreographer, Paxton was interested in the architecture that underpinned movement.

Through his solos – Paxton is involved in "using improvisation as a force to discover architectural and emotional form simultaneously, exploring movement as a reason, not just as an expression of someone's momentary need." (Banes, Writing Dancing in the Age of Postmodernism)

The main characteristic shared by Unreal's triggered scripted sequence and Paxton's score is the listing of actions to be performed by the performers which are set in motion by triggers or cues. In Paxton's work the dancers have a predetermined movement sequence they are to perform which is mainly triggered by the actions of the dancers who go before them. They are not time dependant but

rather take their cues from the activities of surrounding actors. In the following example of a scripted sequence from the Unreal engine (Figure 19), an actor pawn will wait for a specific event (another actor does something, time, end of an action / animation), which will be the trigger to perform the prescribed action or series of actions. So in this example the NPC pawns wait for a trigger whence they move to their allotted starting positions and then wait for cues to perform the next set of actions designated to them. The process is very similar to the method prescribed in Paxton's *Satisfyin Lover*. While the similarities are likely due to expediency; a logical manner in which to organize behavior based on the common human methods of performing one action after another, and responding to an event with a known or even predetermined response if one is available, it nevertheless led me to investigate other common methods of structuring software and choreography mainly based on desired interaction and behavior. The Unreal engine scripted sequence shown (Figure 19) is a high level abstraction in that it is Graphic User Interface (GUI) driven, rather than script driven, tool and is devised in this way to make it easier for non-programming developers to construct bot behavior.

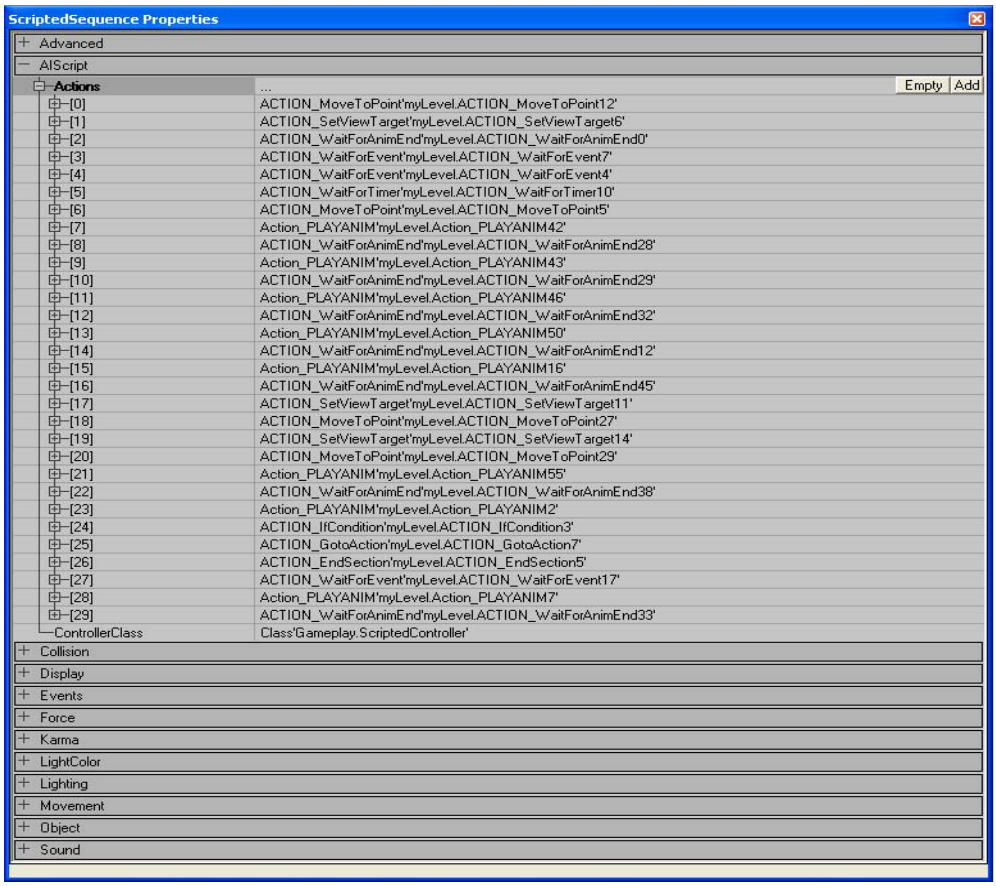

Figure 19: Unreal Scripted Sequence

GROUP A
1. Walk 2/5. 10 second stop. Exit. 2. Cue 10 steps. Walk across. 3. Cue 20 steps. Walk across. 4. Cue #1 pauses. Walk across. 5. Cue #1 pauses. Walk 1/5. Stop 20 seconds. Exit. 6,7,8. Cue 5 steps. enter together. #6 falling gradually behind (at exit be 15 steps behind). 7&8 walk across.

Figure 20: Excerpt from Steve Paxton's Satisfyin' Lover

State Machines

State Machines or Finite State Machines (FSM) are a method commonly used in game engines to define and control behavior. A FSM is usually a model of prescribed behaviours containing a finite number of states or set of instructions, containing a set number of actions to be performed while in that state and able to be entered or transitioned into by some event. In the Unreal engine they are called states and take the form of small components containing blocks of embedded behavior. A state could be seen as an encapsulated piece of behavior that can be called upon when circumstances dictate. A NPC might have a number of states that give scope for the character to respond differently as circumstances change. A finite state machine could also be viewed as an abstract model of a machine with a primitive internal memory. They are often used to give NPCs a primitive AI capability. States are not time dependant in that all members of the state are available at once when that state is entered, there is no linear path through the state. In developing the Unreal Engine a flexible approach to time and states became a core part of the engine, facilitating AI programming.

The major complication in C/C++ based AI and game logic programming lies in dealing with events that take a certain amount of game time to complete, and with events which are dependent on aspects of the object's state. In C/C++, this results in spaghetti-code that is hard to write, comprehend, maintain, and debug. UnrealScript includes native support for time, state, and network replication which greatly simplify game programming. (Sweeney)

Choreographic examples of state machines (that could be said to anticipate the structure of state machines) might be Steve Paxton's *Flat* (1964) (Banes, Writing Dancing in the Age of Postmodernism) and Trisha Brown's *Line Up* (1976) (Banes, Terpsichore in Sneakers 89). In *Line Up*, the dancers have a number of set choreographed sequences that they could potentially perform with a

main sequence and two variations that branch from the main sequence. The sequences can also be performed in retrograde, reversing the arrangement of movements in a sequence. Four dancers perform the sequences according to the instructions given by a fifth performer who cries out the names of the dancers and their instructions; either “Reverse”, “Branch” or “Spill”. The resultant performance is never quite the same as the performers respond to the instructions given them, in turn re-formulating the inter-relationships between the dancers and the movement material. The mental processing power required by the dancers to achieve this performance of known base material dictated by unknown timing and order is more demanding than simply remembering and performing the sequences alone. So too in a software simulation, the artificial intelligence structures required to model this kind of choreographic structure requires more complex structures than to model linear choreography alone. With a state machine, and as in Line Up, the basis of the content is still known, which makes processing somewhat easier. If we proceed to a more extreme example where the movement is unknown until performed and the structure governing the movement is a process of negotiation of the moment of interaction (as in Contact Improvisation) then the processing required of the participants is potentially more complex and may require structures beyond that of state machines and other existing game control paradigms.

Some readers might see an emerging trend in the choreographic works cited as examples thus far. Paxton and Brown were among the artists involved in Judson Dance Theatre which met at Judson Church in New York between 1962 and 1964, and later part of the Grand Union, and are seen as seminal figures in the post-modern dance movement. They were among the leading arts experimentalists of the time whose methods departed markedly from those of the ‘Modern Dance’ period such as Martha Graham, Doris Humphrey and Lester Horton. While the term post-modern might be questionable as it is used in dance history, modern art historians might say their earlier work fits in with the themes of modern art, (Banes, Writing Dancing in the Age of Postmodernism 301) their investigations were nevertheless deeply concerned with the *fundamental structuring* of dance not just with movement invention.

The concept of the "line up" has characterised both format and method in Brown's work. Her major concern has always been to find the schemes and structures that organize movement, rather than the invention of movement per se. (Banes, Terpsichore in Sneakers 86)

It is due to Paxton and Brown's focus on schemes and organizational structure, that I have chosen many of these seminal works as a means to investigate relationships between choreographic structure in dance and organizational structures in software. The movement employed at the time was often very simple, even pedestrian, everyday, non-virtuosic movement was embraced as being equally valid as codified theatrical dance movement. The non-trained body was displayed equally with the trained practitioner. My reason for choosing these works is that the emphasis on structure rather than movement technique helps illuminate the underlying organizational behavior of the works and makes it easier to draw out comparable design choices. It is this emphasis on structure that has led me to question what this implies for the design of software. Could the resultant work benefit if the software and choreographic architectures are purposefully aligned?

My work to this point had tended to use a similar architecture in different works, fashioning different visual outcomes, comparable perhaps to different movements, but based on the same underlying framework. Once I started to see some potential similarities between the organizational structures in software such as the Unreal engine and documented choreographic structures, I decided to make some conceptual design experiments as a way of further exploring these potential connections. As I had begun treating the software environment as a potential dance partner in the performance environment, was interested in game engine technology, and had become persuaded that there may be aspects of the choreographic order that could positively underpin software development efforts, I took a few existing dance works and redesigned them for use with a software partner. In this manner I was able to analyse some of the main choreographic requirements of the dance work and envisage software architectures that might best suit the redesign of the piece for live motion captured performer and digital partner.

The work I have used as examples from Paxton and Brown have largely used pre-choreographed sequences albeit that the nuance of the movement is very performer dependant. The ordering or re-ordering of known movement is similar to the behaviour organization structures occurring in the Unreal game engine. The primary concern of the software is to provide the organizational structures that enable movement to occur without necessarily being concerned with the details of the movements themselves. Different animations can be substituted into the organizational structure at will to produce different variations on character movement. Scripted sequences and state machines in the game environment use pre-recorded animations and as such could be used to organize choreographic structure and in a manner, reproduce a work like Paxton's *Satisfyin Lover* or Brown's *Line Up*. Even substituting synthetic dancers for some of the live dancers might not upset the underlying integrity of the dance too much as the dancers are still performing as individuals with their own logic and scripts that are dependent on the other performers mainly for cues. What of works where the choreographic rules are less defined? Brown's *Theme and Variations* 1972 is a work where the choreographic structure leads to unpredictable or unknown behavior.

> The tension between the mathematical or formal precision of a movement idea and its physical distortion has been Brown's concern in several other dances. In Theme and Variation 1972 one dancer performs a string of movements, and then repeats it over and over as best she can despite attempts by a second performer to interrupt her. The result is a set of variations on the original phrase. (Banes, Terpsichore in Sneakers 85)

The description of this work by Brown reminded me of the similarities with some NPC behaviour in the Unreal game engine. Reactive animation is often generated by combining different animations together through motion blending. A NPC might be walking along when the Player bumps into them. The NPC might respond by still trying to walk but blending in an amount of falling animation away from the Player commensurate with the force of the bump. Another method of causing a NPC to react to a bump or other intervention is to blend an amount of 'Rag-Doll' physics generated movement in with the pre-recorded animation. Rag-doll physics is a method of using a physical simulation to create

an appropriate movement behavior for the virtual bodily mass of an NPC. It is a quite different approach to using pre-recorded animations as the physics simulation creates the unique movement in real-time. For character control in the Unreal engine, there are often a small number of base motions that are augmented by combining multiple motions together to form new variations of those motions, effectively increasing the characters' repertoire. So motion blending might provide a solution to the requirement of taking a set movement phrase being performed by a digital character and modifying the movement based on the motion data of a live dancer. Collision detection could be used to reveal where on the digital character's body the virtual contact was made and an amount of another falling animation could be blended-in based on a reaction to the actions and direction of the live dancer. The following describes the cycle of choreographic tasks involved in *Theme and Variations*.

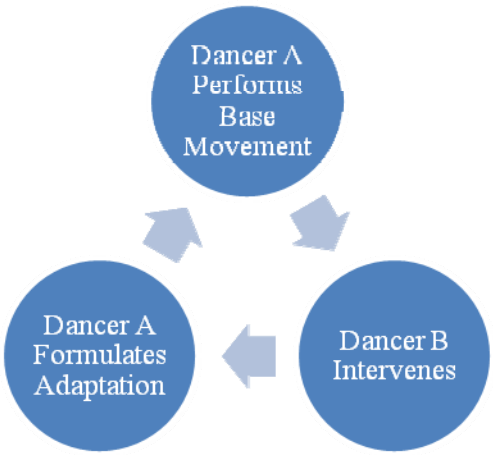

Figure 21: Choreographic Tasks Defined for Theme and Variations

While most of the requirements of Brown's *Theme and Variations* could be superficially replicated in this conceptualized re-staging, the results would possibly have some serious shortcomings. One of the main problems would be in the response of the synthetic character. The animations used as a blend in response to contact from the live dancer's data could not cater for the

varied ways in which a live performer might choose to interfere with their partner, the character could react, but such reactions would tend to be generic and could quickly become repetitive unless the palette of available animations was very large and even then the responses could not be appropriate for all situations, limited, as the choices would be, by the controlling AI's simple 'awareness' of the movement in context. For the programmer's design to account for all possibilities of intervention by a second performer and provide suitable recorded responses would be a painstaking and probably unobtainable task. Use of some instance-generated animation such as a rag-doll based physics response that could be blended into the currently performed animation might be a useful method for introducing a certain amount of organic 'chaos' into the pattern of NPC responses. This chaotic character motion is used in game engines to do things such as generating falling or dying movement for NPCs. The NPC uses pre-recorded movements for most actions, however if an event such as falling down an incline or being shot occurs, the game engine stops using the animations to drive the character and uses a usually simplified physics based simulation to calculate appropriate movement for the character incorporating suitable contact with the environment.

Another possibility would be to use a kinematic system in conjunction with the recorded animation to allow for appropriate modification of the performed actions. The kinematic system could measure the collision from the performer's avatar and adjust the appropriate bones as a modification of the base NPC movements. This could work quite well if the kinematics can achieve reasonable solutions to the interventions. The Unity engine has the basis of a kinematics system that can adapt the current animations to uncertain terrain and obstacles. This could perhaps be adapted to cater for our present problem, however it is not developed enough yet, as it mainly caters for adjusting feet position. However it is a promising direction to pursue. Using animation blending and other tools found within current game technology could provide a partial solution that could passably fulfill the requirements of the choreographic structure in *Theme and Variations*, however the solutions arrived at would be entirely dependent on the mathematical solutions found in the kinematic equations and,

while making the NPC appear more acutely responsive to obstacles, would not entail any real enhancement to its decision making process.

Two current software applications that extend the concept of intelligently modifying motion capture animations to allow characters to adapt their movement are Massive Prime and Euphoria. Massive Prime (www.massivesoftware.com) has been used in the making of large computer generated crowd scenes for films such as Lord of the Rings. Each character or agent in the scene has a software 'brain' capable of making subtle responses to its circumstances through Fuzzy Logic[6]. This allows the Massive agents to provide a finer grain of decision making to the process of co-coordinating the motion captured movement used to animate the characters. So while the characters are still animated largely by motion capture, the subtlety and adaptability with which it is deployed is given finer detail by the individual AI capabilities of the agents.

Natural Motions' Euphoria (www.naturalmotion.com) has been used in games such as Grand Theft Auto IV and Star Wars: The Force Unleashed. It uses techniques that descend from Karl Sim's work on evolutionary generated virtual creatures (Sims). It also uses recorded motion capture sequences as a base for the character motion. The innovative aspect of Euphoria is its use of Neural Networks[7] (NN) and Genetic Algorithms[8] (GA) to find an optimal solution to controlling a biomechanically modeled skeleton. A state machine is then used to apply varying degrees of the motion capture or evolved AI simulation. (Figure 22) The GA method is not necessarily a quick

---

[6] With Fuzzy Logic, decisions can have variable shades of truth as opposed to binary, true/false logic. For example, a car travelling at 65 kph might be deemed, on a scale of 0-1, and characterized as slow, medium or fast, to be travelling 0.8 medium speed and 0.2 fast speed. Thus more than one outcome can be true to varying degrees.

[7] Neural Networks in this context are software components that mimic the behaviour of biological neurons in order to perform certain AI tasks. They are often attempts at simplified models of brain behaviour.

[8] Genetic Algorithms are a type of Evolutionary Algorithm, and usually evolve through the process of running a simulation on a population of potential solutions (or genomes to keep the genetic analogy) in order to reach an optimized solution to a problem. The simulation is often done in several stages, taking the best solutions from one stage and running them through the simulation again to further refine subsequent 'generations'.

solution though, it takes time to evolve a stable working solution and in the case of Euphoria, this tacit 'tuning' process is done outside the game engine and only later inserted when a robust solution to a particular set of game environment circumstances is arrived at. It is interesting to note that the task cycles for Brown's *Theme and Variations* and Euphoria's *Dynamic Motion Synthesis* (DMS), as the publishers Natural Motion like to call the process, are extremely similar, suggesting they could be a good match between the desired choreographic and software constrained outcomes (Figure 21) and (Figure 22).

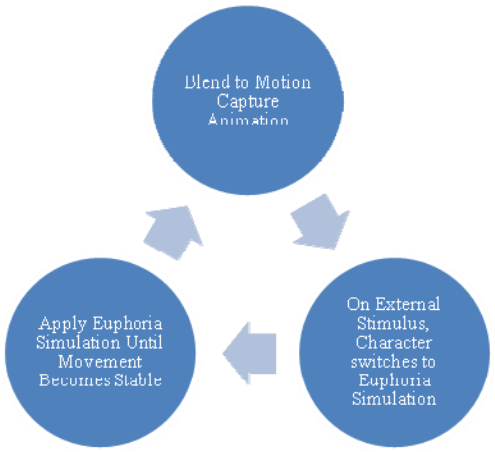

Figure 22: Euphoria Animation Control Cycle

While the GA based solution used in Euphoria could well yield acceptable results to our choreographic task, the requirement of evolving a solution over several generations may not suit a live performance situation where a solution is required immediately in response to the dancer's input. What our digital character ideally requires is a way of developing appropriate responses to the nature of the interruptions in an autonomous fashion. If we were to develop a sufficiently sophisticated dance partner, it needs to be able to assess the quality and magnitude of the motion from the live partner and generate an appropriate response. There may be patterns to the interventions that the character could learn from, which would then aid in formulating variations. This is where the AI in the game engines I have used to date reach the limits of their current capabilities. The AI required to formulate and animate our character appropriately is very complex and requires a more complex solution, one that

can perform intelligent actions in such a dynamic environment. (Russell) By dynamic environment I refer to the unpredictable nature of the live motion capture data in comparison to the normal range of pre-recorded actions available to a Player character and NPC. Pre-determined responses are likely to be inappropriate to unforeseen movement activity.

One approach would be to use a more sophisticated form of AI capable of assessing an unknown terrain (a second body) and reacting accordingly by appropriately animating the character's skeletal structure. One possible method is suggested by Liu (Liu, Autonomous Agents and Multi-Agent Systems 73) in response to a desire to create digital, synthetic agents that can autonomously self-animate, adapt to their environments, and learn new behaviours to attain some specific goal. Liu's method is to utilize a multi-agent approach to the problem of addressing an unknown terrain.

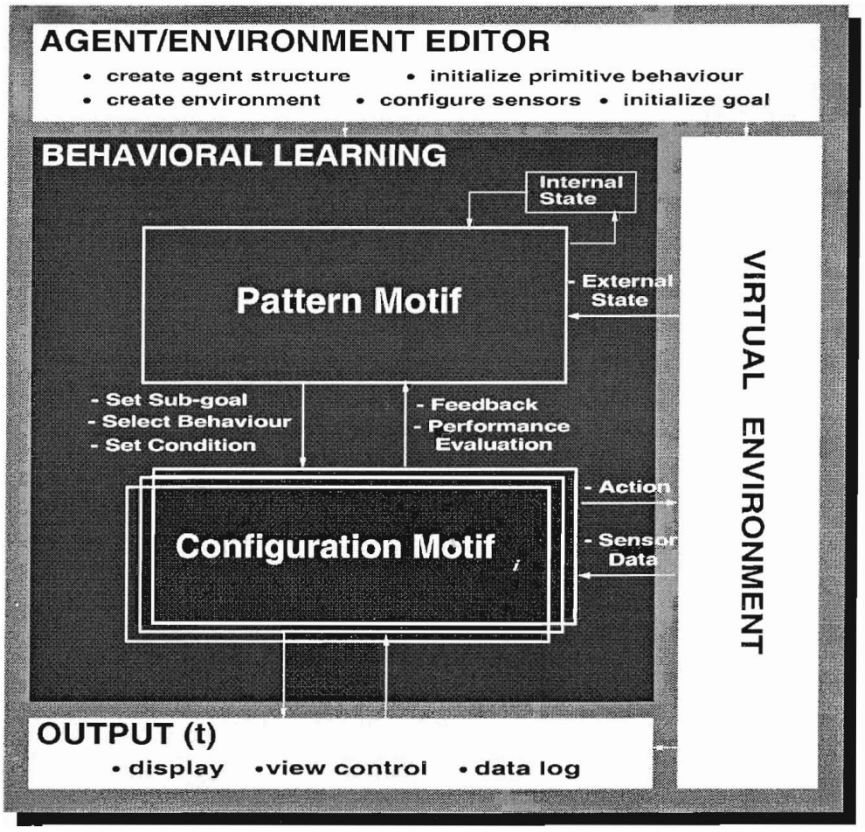

Figure 23: A Schematic Diagram of the System for Building Life-Like Agents with a Behavioral self-organising capability. (Athlete) (Reprinted from LIU 2001 98)

Towards this end Liu has created an autonomous animated character called the Athlete Agent. Athlete seeks to navigate around an unfamiliar 3D environment, adapting to obstacles and unfamiliar terrain. Athlete must seek solutions allowing it to overcome the obstacles and continued navigation of the environment. One of the key elements of Athlete and of most research into agent-based software is the ability for the agent to learn from its experience. To this end Athlete employs what Liu terms Motifs, interweaving software threads that can be combined or clustered to form a self-organizing computational unit. (Liu, Behavioural Self-Organizaion in Lifelike Agents 254) The individual motifs are capable of behavioural self-organization towards some specified goal as well as being able to take into account state information from other motifs in the cluster and passing their own current states on. The motif-based behavioural organization will reach a state of dynamic equilibrium, whereby no drastic changes are required of the states or any of the motifs, and a current, optimal solution will have been arrived at. Athlete employs two motifs termed Pattern-Motif (P-Motif) and Configuration-Motif (C-Motif) that interact to provide solutions to the current problem. P-Motif engages with the environment and the current Athlete internal states to generate mappings of each. (Figure 23) C-Motif optimizes itself to arrive at an optimum solution based on the mappings provided by P-Motif and then returns its decisions as feedback to P-Motif and the process continues. While the implementation of such a system has been beyond the current scope of the project component of this project, the devices it suggests are very promising. Essentially Athlete's decision making process is governed by the negotiation of two interwoven organizing threads that have autonomy to pursue their own goals but also share the results of their processes resulting in a higher level solution between them. One Motif surveys the environment and the current state of Athlete and makes possible suggestions. The second Motif surveys the suggestions and chooses the solution which it deems most fit and lets Motif 1 know of the choice so it can update its next survey. One thing to note about this approach is that there is an emphasis on the behavior and interaction of and between the Motifs.

*It is now widely recognized that **interaction** is probably the most important single characteristic of complex software*. (Wooldridge 7)

Over the years, a wide range of software engineering paradigms have been devised (e.g. procedural programming, structured programming, object-oriented programming, design patterns, application frameworks, and component-ware) to deal with the increasing complexity of software applications. Most software engineering paradigms are unable to provide structures that make it easier to handle this complexity. Consequently a lot of research has now been directed toward treating computation as a process of interactions. (Biswas 3)

Could this approach, treating computation as a process of interactions, be adapted to solve our choreographic requirements? Let us look at the sequence of events for Athlete. Athlete proceeds in a state of equilibrium (walking) until an obstacle is encountered. Through its Motif architecture it assesses the obstacle and its current state, then attempts a solution to overcome the obstacle until it can return to a state of equilibrium again. Likewise, in *Theme and Variations* our dancer performs her pre-determined movement until she encounters an obstacle (in our case, the physical intervention of another dancer), she assesses the obstacle and the state of her body in motion and formulates a solution to overcome the obstacle and return to the base movement material. In terms of a digital performer and a live motion captured performer, a similar process unfolds. The digital performer proceeds performing predetermined movement until an obstacle is encountered. It assesses the environment (the topology of the motion data from the live dancer) and its own state (the current movement state) and formulates a solution in order to return to the original movement. The approach taken by Liu is very attractive for a number of reasons; it allows us to assign the choreographic tasks required of the dancers, in terms of decision making, to the motif agents; it allows us to divide the work into modular components that have some autonomy yet closely inter-relate; and it gives the work scope to adapt to new circumstances or possibilities that would be extremely difficult to account for in a more traditional, scripted software approach.

From the perspective of the choreographer, the first point is perhaps the most interesting, the ability to think of interacting software components in terms of behavior, and here in terms of choreographic behavior. Liu's work belongs to the rapidly expanding area of agent-based software and in particular multi-agent systems. Conveniently, Athlete also uses a primitive skeletal system, which while focusing on lower body locomotion, could be expanded to cater for a full body skeleton as currently used in the present projects. In order for the Motifs to gather the required information for the internal and external states on which behavioural choices are based, several virtual sensors are mounted on the body of Athlete. Body sensors are installed on the body to prevent body parts colliding, Internal sensors are used to detect the agent's internal state and External sensors are mounted to detect the presence of virtual objects in the environment that might present as obstacles. The Body and Internal sensors could be viewed as being analogous to the human proprioceptive system while the external sensors could be viewed as being akin to the human peripheral senses. Viewing the sensor system in this way may help clarify the possible system architecture as it puts it in the context of a known existing system which can be clearly visualized. The methods of surveying the body and the environment in order to arrive at suitable solutions would have a similar basis for both dancer and software.

In terms of the practical development of such a work, based on an agent architecture, the Motif agents could be written as plug-ins for a game engine environment which would have its advantages and disadvantages. Having the agents running inside the game environment would allow the work to be easily distributed and run as a single application, however the agents would tend to be peculiar to that game environment, although they could send information out of the engine if required.

Liu's Athlete uses quite a different approach from those encountered within the Unreal game environment. Using state machines or triggered sequences, the solution arrived at will be a pre-defined solution as the system jumps to what is evaluated as an appropriate state. A state machine only acts upon inputs, outputs are determined by the input and the resultant state. With the approach suggested by Athlete, the previous assessments made by the evaluating motif-agents become

incorporated into next iteration of evaluation, the output has a direct influence on the input as it is looped back into the system, allowing scope for more varied solutions than from sequential or state-based solutions. Within the Athlete approach, the emphasis is on negotiation between the agent components regarding appropriate interpretation of the data with a view to how they can interact to achieve a suitable behavioural outcome.

Another solution to dynamic behavior in NPCs is suggested in RiskMan (Kavakli 357). The agents in RiskMan (Figure 6) run independent from each other in their own processes, and send the results to the game engine over the network. There are some advantages to this approach; the components can be developed in any language or mix different development platforms and styles of programming whereby the components can be incorporated with changes and updated, completely independent of other components and the components could reside on separate servers or migrate from one machine to another as required rather than having to run on the same machines. This last point borrows from research into mobile agent architectures. There are a number of uses for this distributed, mobile architecture other than encapsulating the agents into the game engine. One of the key themes of my work has been the collaboration of data streams coming in over a network into a virtual environment. In a mobile, multi-agent based approach, each motion data stream might come with its own agent interface that can analyse the data and provide appropriate information to other agents and to the environment as a whole in the pursuit of collaborative use of the motion streams. RiskMan provides an example of such an architecture, the simulation agent, scripted agents, trainee agent interface agent and communication agent interact to provide an optimal simulation experience for the trainee through control of the narrative and behavior engines, the results of which are then passed onto the game engine for the display of virtual characters with whom the trainee interacts to enact the simulation experience. The characters in the RiskMan game environment use only prerecorded motions, however the work undertaken by Liu provides a means of inserting appropriate dynamically generated responses into the simulated environment.

Integrating aspects of RiskMan and Athlete together with the motion playback and blending capabilities of a game engine could provide an architecture capable of providing an entirely or partly simulated solution to the choreographic requirements of Brown's *Theme and Variations* in as far as a digital character can provide a scaled-down version of the depth of human motion. The choreography is based largely on known movement material modified from an outside source.

Towards Agent-based Contact Improvisation

How would we deal with a choreographic situation where the movements are not predetermined and the negotiation between the two performers is the source of the movement invention? One such choreographic situation would be the case of contact improvisation. The invention of contact improvisation, certainly the term itself and its early propagation, has been ascribed to Steve Paxton. (Novak 10) In contact improvisation the emphasis is not so much on the choreographic shaping of movement phrases or narrative or emotional expression but rather on the physical dialogue between two dancers and the interaction resulting from shared touch, weight and contact. As a form to be emulated as a live and digital partnership, this free form, improvisational type of choreographic structure, requiring decisions to be made from moment to moment, is far harder to emulate in a software system than when using prerecorded or known movement material as a base. It is in such a dynamic system, where decisions must be constantly negotiated and solutions generated, that a distributed multi-agent architecture approach may be the only real software solution. (Liu, Autonomous Agents and Multi-Agent Systems 90) While certainly associations between the motion of a live dancer and a digital partner could be made in most software systems, to achieve a solution even partly comparable to that which two people would arrive at would be extremely difficult in traditional approaches to software development. While the combined agent approaches of Liu and Kovakli would be a means of structuring the heavy AI demands of emulating contact improvisation processes in software, adequate means of analysing the live dancers' motion stream and the digital response are also required.

The approach taken in the Dance Education System (Leung) presents a means of analyzing the data streams and supplying possible responses. In the Dance Education System the skeletal joint rotations from the student's movements are measured and compared to the teacher's performance of the same movement sequence. The degree of closeness of the student's movements to that of the teacher are gauged and feedback is presented to the student on areas of the movement that are close to that of the teacher and areas that might need more attention. Using this method it is also possible to recognize predefined gestures when they are performed by the student. With a work like Brown's *Theme and Variations* this approach might be adequate. Because we are starting with known base material, the respective agents could use the techniques employed by Leung et al to analyse the current desired state of the movement and the state of the second, live dancer's motion and formulate a variation on the original movement. A second set of agents could assess the current modified state of the digital character, along with the desired base movement and formulate a strategy to return to this known base movement. However, to account for two constantly changing streams of motion with no predefined movement references, as in contact improvisation, would require an extra layer of behavioural interpretation.

A Behavioral Language for Dance and Software

We have a curiosity about body organization and the physics, subtle and gross, of bodies moving through contact, primarily in duet, but inclusive of other configurations. (Body-Research)

If you're dancing physics, you're dancing contact. If you're dancing chemistry, you're doing something else. (attributed to Paxton 1987)

…the aesthetic of contact improvisation was largely conceived of as an appreciation of movement based on "survival" and on the "natural" and 'honest" outcome of playing with weight, momentum, and gravity, an implied contrast to more "artificial, contrived" dance forms based on ideas or aesthetic concepts. (Novak 181)

I am not necessarily a big fan of contact improvisation as a theatrical dance form, I find its merit is in the doing rather than as a spectator. However the emphasis on the physics of the bodies is useful in our process of modeling behavior as there already exist examples of physical simulation tools within 3D simulation environments. Having to find behavioral solutions based on conceptual and aesthetic choices as well, would greatly increase the level of difficulty. This emphasis on the physics of bodies is another reason contact improvisation makes a useful genre in which to explore a synthetic partner's potential behavior.

By incorporating a physics-based animation system such as DANCE [9] (see above) into a game engine framework we would have the tools to investigate solutions to the problem of generating dynamic movement responses by a digital partner to a live dancer's movement data. Borrowing also from Athlete and RiskMan, an agent-based architecture could be employed to analyse the current state of the dancers' representations and then use the dynamic physics-based character animation engine (DANCE) to generate appropriate movement responses for the synthetic dancer. The methods employed in DANCE could be used in place of the narrative engine in RiskMan, or by the configuration-motif in Athlete to generate appropriate physical responses for the synthetic dancer.

As mentioned before, one of the key requirements of the Athlete agent is that it can autonomously self-animate, adapt to its environments, and learn new behaviours to attain some specific goal. (Liu, Autonomous Agents and Multi-Agent Systems 73) This designation of, and attempt to attain some specified goal, is a key component of agent design. Thus the goals set for the agent are of paramount importance to the eventual outcome they produce. How do we choose the goals for the agents? A shared set of goals between the agent system controlling the actions of the synthetic character and the live dancer could be a starting point. This would allow the choreographer or designer to prescribe the same set of tasks to the live dancer and the software agents controlling the synthetic character's movement behavior. This goes a step further from using similar organizational structures to develop

[9] http://www.arishapiro.com/researchportfolio/DANCE/

solutions to choreographic tasks, such as using a state machine controlling pre-recorded and dynamically generated animations, to meet the requirements of *Theme and Variations*, to using a common definition of goals to be attempted by the live performer and agent. This development of a common syntax, able to be applied to both the live performer and the synthetic, agent-driven partner has ramifications for the potential ease with which a choreographer might interface with and direct, complex software agents.

An example of common goals which could be set for the different parties might be taken from a description of some possibilities inherent in contact improvisation suggested by Paxton in a 1975 article. (Novak 182) After stating that contact improvisation "as a social system" comprises different combinations of the elements active (A), passive (P), demand (d) and response (r), Paxton continues with the following description:

One [person] may lift the other (Ad & Pr) [Active demand and Passive response]. One may fall so the other must catch (Pd & Ar) [Passive demand and Active response]. One may attempt to lift and find the energy translated so s/he is lifted (Ad & Ar) [Active demand and Active response], etc.

From this truncated description a set of goals could be formulated; determine whether you are in an active or passive role, determine whether you are in a position of demand or response, and based on the current input from the physical simulation formulate an appropriate action.

This basic framework gives us a means of creating meaningful associations between the two constantly changing and somewhat unpredictable movement streams of the live dancer and synthetic character. The determination of whether the (live or virtual) dancer is in, or should attain, an active role and is therefore required to instigate a new behavior gives our system a means of determining when to instigate purposeful action. The choice of action would be partly determined by the current state of the virtual bodies in the simulation and partly by the types of actions available.

In Athlete the actions formulated by the agent motifs are grouped under generic conceptual groupings; Walk, Hop, Jump, Run and Squat. These become definitions for primitive behavioural patterns. This reduces the complexity of the architecture to behavioural states and subsequent interaction with the environment. This would be a useful method to retain with the choice of behavior pattern primitives also drawn from Paxton's description. Lift, Fall and Catch could be added to basic locomotion patterns. Along with the goals of determining their current and potential active or passive status, and the possible need for instigating action, this would give our agents a means of determining when an active response is necessary and the ability to instigate a behavior type in keeping with the shared expectations of the live and synthetic dancers based on a set of shared choreographic goals.

Conclusion

During the process of developing new work I have sought means of integrating the digital and human components of a performance into a cohesive whole. There are many parameters to consider in achieving this; choreographic design, stage design, projection design, visual design - both physical and virtual - and the design capabilities of the digital tools used in the making of the performance to name a few. In this research I have looked mainly at the role of choreographic and software development, in creating a performance. In earlier works the two were often interdependent, e.g. if there was no movement there was no response in the visual environment, and while this might seem like a good thing, this caused a certain lack of flexibility that often restricted either or both aspects from developing beyond an initial stage. I have looked here at whether the two elements can co-exist in other more intrinsic ways that would be beneficial for the resultant work. I asked myself the question whether creating shared structures or goals for both the choreography and software might lead to beneficial results. I have come to the conclusion that this is indeed a path forward in seeking to integrate the live and digital environments. The use of appropriate software structures aligned to the type of choreographic structure in use can be taken into consideration to efficiently order the movement behavior of the synthetic characters in order to achieve the desired choreographic tasks.

For relatively dynamic choreographic structures an agent-based approach may provide a solution to formulating movement solutions for synthetic characters. An agent architecture focused on goals, interactions, responses and behaviors provides an interface to software tools that has similar attributes to interactions found within many choreographic works. From the perspective of a choreographer this might transpire as the ability to pass the same tasks or goals to dancer and software environment and require them to negotiate a suitable outcome. The potential to conceive choreographic task development and software development as developing from the same basis and towards the same

inherent goals means that they need not necessarily be considered as separate components requiring different approaches and trains of thought.

Bibliography

Banes, Sally. Terpsichore in Sneakers. Boston: Houghton Mifflin Company, 1977.

—. Writing Dancing in the Age of Postmodernism. Hanover: Wesleyan University Press, 1994.

Biswas, Pratik K. "Toward Agent-Oriented Conceptualization and Implementation." Lin, Hong. Architectural Design of Multi-AgentSystems: Technologies and Techniques. New York: Information Science Reference, 2007. 1-25.

Body-Research. Contact Jam. 2009. 10 01 2009 <http://www.bodyresearch.org/contactjam.shtml>.

Bradshaw, Jeffrey. Software Agents. Cambridge, Massachusetts: MIT Press, 1977.

deLaHunta, Scott. The Dimensions of Data Space. 29 01 2003. 10 11 2008 <http://www.sdela.dds.nl/mcrl/index.html>.

igloo. igloo. 1 January 2000. 11 November 2008 <http://igloo.org.uk>.

Kaiser, Paul. "Paul Kaiser of Riverbed Interviews Michael Girard and Susan Amkraut." 13 04 2002. Ohio State University. 24 10 2008 <http://design.osu.edu/carlson/history/PDFs/girard-amkraut.pdf>.

Kavakli, Manolya. "RiskMan: A Multi-Agent System for Risk Management." Lin, Hong. Architectural Design of Multi-Agent Systems: Technologies and Techniques. New York: Information Science Reference, 2007. 356-376.

Laurel, Brenda. "Interface Agents: Metaphors with Character." Bradshaw, Jeffrey. Software Agents. Cambridge, Massachusetts: MIT Press, 1977. 67-77.

Leung, Howard and Chan, Jacky and Tang, Kai-Tai and Komura, Taku. "Ubiquitous Performance Training Tool Using Motion Capture Techmology." 1st International Conference on Ubiquitous Information Management and Communication. Seoul Korea, February 2007.

Lin, Hong and Yang Chunsheng. "Chemical Reaction Metaphor in Distributed Learning." Proceedings of the International Conferenace on Industial & Engineerning Applications of Artificial & Expert Systems (IEA/AIE 2004). Ottawa, Ontario, Canada: NRC 46551, 2004.

Lintermann, B. Haffner, N. McManus, T. Timelapses. 2000. 21 November 2008 <www.timelapses.de>.

Liu, Jiming. Autonomous Agents and Multi-Agent Systems. Singapore: World Scientific, 2001.

—. "Behavioural Self-Organizaion in Lifelike Agents." Proceedings of the Second International Conference on Autonomous Agents. Minneapolis, 1998. 254 - 260.

Murray, Janet. Hamlet on the Holodeck. New York: The Free Press, 1997.

Novak, Cynthia J. Sharing the Dance: Contact Improvisation and American Culture. Madison, Wisconsin: University of Wisconsin Press, 1990.

OpenEnded Group, The. how long… / 2005. 2005. 20 January 2009 <http://www.openendedgroup.com/index.php/artworks/how-long/>.

Russell, S.J., & Norvig, P. Artificial Intelligence: A Modern Approach (2nd Edition). New Jersey: Prentice Hall, 2003.

Sims, karl. "Evolving Virtual Creatures." Computer Graphics (Siggraph '94 Proceedings). Siggraph , 1994. 15-22.

Sweeney, Tim. "UnrealScript Language Reference." 21 December 1998. epicgames. 14 January 2008 <http://unreal.epicgames.com/UnrealScript.htm>.

Van de Velde, W. "Cognitive Architectures - From Knowledge Level to Structural Coupling." Steels, L. The Biology and Technology of Intelligent Autonomous Agents. Berlin: Springer Verlag, 1995. 197-221.

Wooldridge, Michael. An Introduction to MultiAgent Systems. Chichester: John Wiley & Sons Ltd, 2002.