A Realistic Distributed Interactive Application Testbed for Static and Dynamic Entity State Data Acquisition

Damien Marshall[ϕ], Aaron McCoy*, Declan Delaney[ϕ], Seamus McLoone*, Tomas Ward*

[ϕ]*Department of Computer Science,*
NUI Maynooth
Maynooth, Co. Kildare
E-mail: damienm@cs.may.ie

** Department of Electronic Engineering*
NUI Maynooth,
Maynooth, Co. Kildare
E-mail: tomas.ward@eeng.may.ie

Abstract – **Scalability is an important issue for Distributed Interactive Application (DIA) designers. In order to achieve this, it is important to minimise the network traffic required to maintain the DIA. A commonly used technique to reduce network traffic is through short-term entity dynamics extrapolation. However, this technique makes no use of a priori information regarding entity dynamics. We have been developing methods to employ this information through a number of techniques, primarily statistical in nature, which have shown great promise in constrained experimental environments. The main tenet of our approach is that user behaviour in real DIAs follows patterns, and through acquisition, analysis and exploitation of these patterns, a reduction in network traffic can be achieved. In this paper, we report on our development of a realistic DIA based on an industry standard SDK in which we have implemented data acquisition routines that allow us to do this. Results are presented for trial runs using the system. These results clearly exhibit patterns of user behaviour consistent with our previous research and suggest that the exploitation of this knowledge can help reduce network traffic.**

Keywords **–Distributed Interactive Applications, Entity State Modelling, Human Behavioural Modelling.**

I INTRODUCTION

A Distributed Interactive Application (DIA) is a distributed virtual reality system through which individuals can share information via individual and collaborative interaction with each other and the environment [1]. When the fast-paced nature of these interactive applications is coupled with the potentially large number of concurrent users, large volumes of update packets tend to be generated. This, along with latency, adversely affects the scalability of the system.

If latency were a constant factor in such an application, provisions could be made within it to compensate for the delay. However, the variation in latency times, known as jitter, makes any solution to the latency problem even more difficult to implement. Other problems in DIA design, such as graphics rendering or physics simulation, may be attributed to current technological limitations, and possibly solved with the advent of faster hardware. However, the same does not apply to network delay. It is a common misconception that an increase in bandwidth can eliminate the problems of latency and jitter entirely. Due to the physical properties of network packet transmission, delays are ultimately insuperable [2].

Therefore, rather than combating these problems directly, methods exist to compensate for their effects on the user and the application. One popular method is a client side prediction contract mechanism known as dead reckoning [3].

However, short-term predictive methods such as dead reckoning do not make use of a priori information regarding entity dynamics. Techniques employing user behaviour analysis, such as strategy-based models, have been shown to perform better than short-term models in constrained experimental environments [4, 5].

Techniques such as dead reckoning and strategy modelling have been tested in applications that demonstrate their effectiveness but may be considered unrelated to real world DIA scenarios [5, 6, 7]. There is a need for a DIA testbed that can accurately recreate the performance, bandwidth consumption, and user experience provided by current popular DIAs, such as the well known computer game Unreal Tournament [8].

This paper will focus on the development of such a testbed for the comparison and testing of latency and network traffic reduction mechanisms. Created using an industry-proven game engine, it contains elements comparable with modern day game systems, along with comprehensive data acquisition routines. Preliminary data gathered by these routines clearly show player movement patterns that underlie the motivation behind the strategy model approach [4,5].

Section two gives a brief overview of our chosen research platform, Torque. In section three, the test scenarios and environments are detailed, and our results from these tests are discussed in detail in section four. Finally, section five provides some conclusions on these results, and gives some insight into our future work in this area.

II TORQUE

a) A Brief History

The technology behind Torque is a modified version of that which featured as the game engine behind the award winning games Starsiege, Tribes, and Tribes II, all of which were published by Sierra Games [9]. In 1998, former employees of Dynamix, the developers of Tribes II, began negotiations with Sierra for the rights to the source of this engine. With the rights secured, they formed a company known as Garage Games, and began to license the engine to both amateur and commercial developers [10].

b) Torque as a Research Platform

One of the most important issues at the beginning of our research was that we would have to pick a platform that would allow us to conduct our research as efficiently and as thoroughly as possible, and yet still allow for a great deal of flexibility.

We considered using the Unreal Tournament engine. Although it has a large user base, and a powerful scripting language, we found the non-availability of source code to be a major disadvantage. We also deliberated over the use of the dated, but open source, Quake engine [11], but found that the lack of support and amateur development community would hamper our ability to fully utilise the more powerful aspects of the application

Finally, we decided that the Torque engine would fully meet our needs. As mentioned above, the engine is completely open source, and has a proven track record with the popular Tribes series of games. The fact that it is open source would allow us to potentially develop any category of application using the engine as a base.

It features a powerful scripting language that facilitated the rapid development of our test bed application, with minimal editing of the C++ source code. In addition the source code was well structured, clearly commented and easily understandable. This makes the process of adapting the application to our research easier.

The Tribes series were designed with play across a network in mind. Because of this, the network code, which is of most interest to our research group and is among the most robust and reputable in the industry.

Due to these factors, a group of amateur developers, which now number in the tens of thousands, have formed a community around the engine, leading to strong customer support, useful tutorials and a very active forum service at the Garage Games website. The presence of a supportive development community was a very important consideration in our choice of platform.

In this section we have introduced the Torque Game Engine, and detailed our reasoning behind choosing it as our main research platform. Next, we discuss the test scenarios and environments that we conceived and developed using the features of the Torque engine.

III TEST SCENARIOS

Two separate testing scenarios were constructed in order to acquire data related to two different categories of strategy model, known as static and dynamic strategies. The static test dealt with single players and a static goal position, and the dynamic test involved multiple players with ever changing main goal and sub-goal positions.

In both cases, users were given a questionnaire in order to assess user experience with such gaming environments. They were also presented with a brief explanation of the task that they were being asked to accomplish within the environment. They were then allowed a single run inside an empty training environment in order to become accustomed to the control interface. When the user indicated that they were satisfied with the interface and the test rules, the test began and the user was placed inside the relevant test environment.

An overview of both test scenarios, along with their respective environments, is now presented.

a) Static Test

The static test allowed for single players only. Both the starting position of the player and target position were invariant. The goal in this case was to navigate from the starting position to the target position in the shortest time possible. When the user reached the target position, they were notified of the time taken and the test was then restarted.

The static test featured two individual environments that aimed to present different circumstances to the user. The first environment consisted of a single well-defined route from start to target position. In order to disorientate the user, the environment featured an apparent dead-end route and multiple buildings – a plan view of the environment is shown in Figure 1.

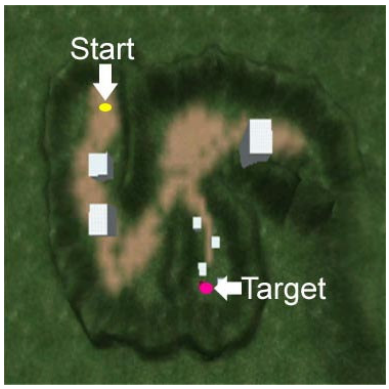

Figure 1: Static Test - Environment 1. The white rectangles represent towering obstacles.

The second environment consisted of two, well-defined routes from start to target position, with one route initially more apparent to the user. These two routes were comparable in the time taken to navigate. In addition, an apparent dead-end route was added to provide some user disorientation during navigation. This second static test environment is shown below in plan in Figure 2.

Figure 2: Static Test - Environment 2. There are two obvious strategies to reach the target.

Data was recorded for the user on each test attempt. This consisted of the user's positional data within the environment (X, Y, and Z co-ordinates), which was recorded at regular intervals, along with any actions that the user performed, such as pressing a keyboard control to move forward or backward.

b) Dynamic Test

For the dynamic test a simple game was designed that allowed multiple users to play against each other in a single environment. The main feature of the game is referred to as the 'tag'. Each game featured a game timer, initially set to 150 seconds. The goal of the game is to be the player holding the tag when the game timer reaches zero. When a player is holding the tag, it is not possible for them to shoot. The game time decrements only when a player has control of the tag. Other players make the tag holder drop the tag by disabling them with their weapon. Each player has an on-screen radar that shows the position of any players within close proximity.

In addition to the simple rules defined above, the environment features four 'health-houses' that correspond to North, South, East and West – see figure 3. These houses allow for the full regeneration of a player's health. However, only one house is active at any one time. Entering an active health house results in the random selection of another house to become active immediately.

The game randomly chooses the active health-house at regular intervals and each player is sent a message informing them of its location. These health houses allow for the generation of sub goals during the test, as players scramble to reach the active health house before they are disabled.

Figure 3: Dynamic Test Environment - 'Bird's Eye' view

For each test, a single game server was created. Each player then connected to this server as a client. As with the static test environments, data was recorded for each game that was played. Each client recorded their own information, namely the player's positional data (X, Y and Z co-ordinates), along with any

actions that the user performed during the game. At the end of each game, each client recorded the total length of time that the user was in possession of the tag throughout the game, and whether or not they were the eventual winner. In addition, each client recorded his or her local representation of the other players that were playing the game. In effect, we are capturing each player's local view of the environment and of the other players within that environment.

Figure 4 details an in-game shot of the game in action, along with the player's Heads Up Display (HUD).

Figure 4: First-person view, showing various components of the 'HUD'

c) Summary

This section has detailed the two main tests from which data was gathered. A test featuring a static user goal and also a dynamic test featuring multiple players was detailed. The dynamic test accurately recreates real world game play in terms of rules and user experience, which should lead to data that is more consistent with that of current genre leaders.

The next section analyses the preliminary data acquired from these two tests, and provide some insight into how these observations reinforce current research in this area.

IV RESULTS

a) Static test results

Six independent tests were carried out using the approach detailed in section three and the data was collected accordingly. Test subjects of varying age, sex and gaming experience were chosen, so as to provide a broad range of data.

As a preliminary analysis of the data, we examined the relationship between the time taken for each attempt for the six test subjects and the number of attempts taken, the results of which are shown in Figure 5. From this plot, we can see that, in general, as a user spends more time in an environment, the time taken to achieve the goal decreases steadily. Notice also that during the initial attempts of some test subjects, a certain amount of variation occurs in the goal achievement times. These fluctuations can be attributed to exploration carried out by the test subject.

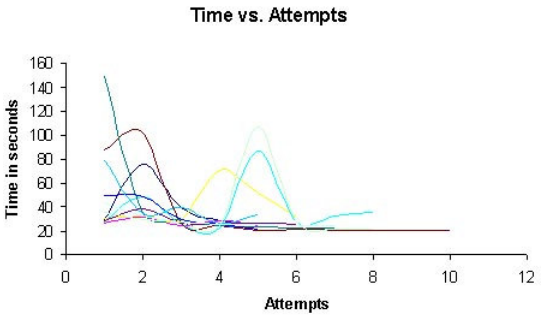

Figure 5: Time vs. Number of Attempts

Figures 6 and 7 detail plots of the trajectories of all user attempts in both test environments, with an outline of the maze superimposed on each plot. Observation of these plots show that the majority of paths correspond to single strategies. Even in sub-optimal attempts, test subjects still adopt similar patterns. This observation underlies the motivation behind the strategy model approach [4, 5].

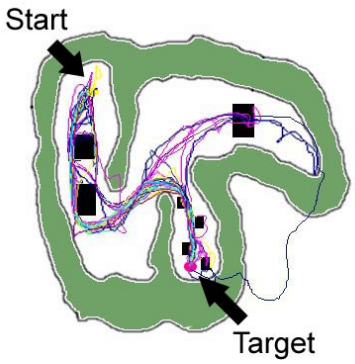

Figure 6: Complete results of environment one showing all trajectories for all users.

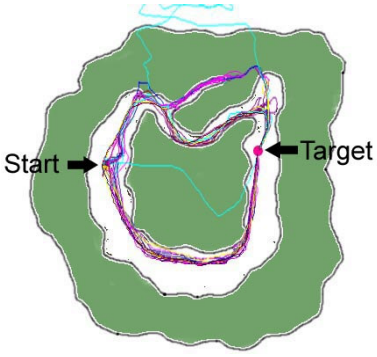

Figure 7: Complete results of environment two

In order to provide a clearer image of general user behaviour in each environment, figures 7 and 8 consist of the fastest result of each of the six test subjects, with the outline of the relevant environment superimposed on the plot.

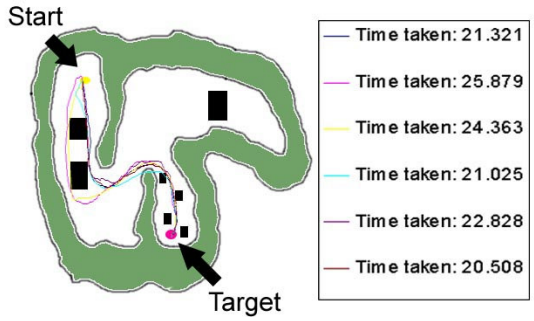

Figure 8: Plot of trajectories corresponding to the best times for environment one

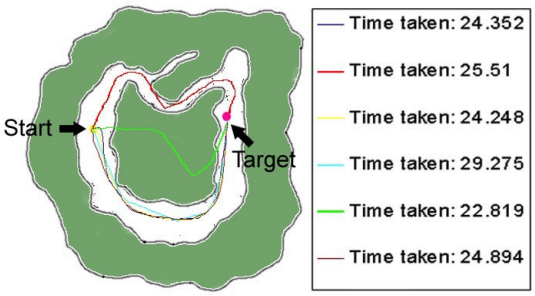

Figure 9: Plot of trajectories corresponding to the best times for environment two

From these figures we can see that users eventually adopt a steady-state strategy to reach their goal. From observation of the subjects as they took the test, we can attribute the difference in times to lack of familiarity with the control scheme, although this did not have much effect on the paths chosen by each test subject.

From analysis of Figures 5-9 the following observations can be made:

1. In an environment with a set goal and starting point, users will quickly learn the layout of the environment. Several exploratory attempts are usually required in order to achieve this.
2. Over time, predominant strategies are adopted in order to reach the goal. Players converge to recognizable steady-state strategies.
3. During the initial attempts of the user in the environment, a level of exploration occurs, leading to erratic test times. Although user behaviour appears random, comparison of trajectories from different users at this stage in the test shows that similar strategies are adopted.
4. Once initial exploration is finished, and the goal is achieved in a reasonable time, users tend to follow the same trajectory repeatedly, and attempt to improve the performance on this path, rather than search for new paths to the target.

b) Dynamic test results

Once again, six tests were carried out using test subjects of varying age, sex and gaming experience. A number of trials were conducted with pairs of users and the data recorded as before. Figures 10 and 11 shows the data collected for two different trials for one pair of users.

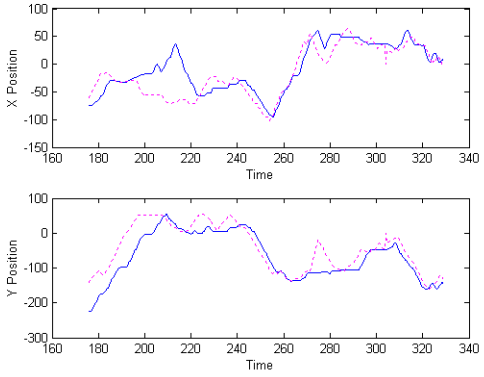

Figure 10: Entity positional state (x,y) versus time, trial 1. The solid and dotted lines represent users 1 and 2 respectively.

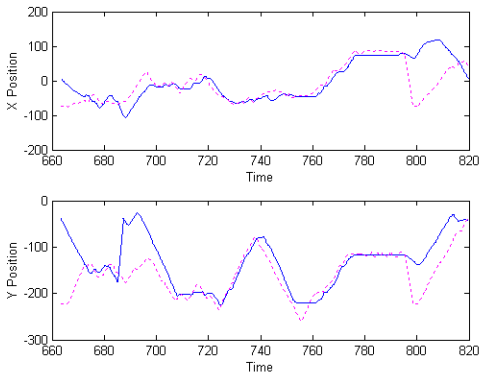

Figure 11: Entity positional state (x,y) versus time, trial 4. The solid and dotted lines represent users 1 and 2 respectively.

These plots clearly illustrate that the interaction between the two players results in vastly different behaviour to that of the static case. Here, it is intuitively obvious that static spatial patterns will not emerge due to the changing nature of the goal, viz., finding the other player. Consequently it is difficult to discern patterns of behaviour from a purely spatial representation of the data without reference to its temporal properties. Hence we have plotted the data

with time as the abscissa. An interesting observation that can be made in the dynamic case is the similarity between the trajectories followed by the two users. This clearly reflects the pursuit of the tag-carrying user by the second player. This is an obvious and apparently universal strategy in this game scenario. Such patterns in dynamic scenarios could be utilised in further development of entity state update mechanisms.

V CONCLUSIONS AND FUTURE WORK

In this paper we have described a DIA testbed that accurately recreates realistic game situations in terms of user experience and graphical, aural, and network performance. These properties are important as it leads to the collection of data that is closer to a real world game scenario and therefore more useful for accurate research.

Using the data collection routines of this testbed, we have acquired results that exhibit patterns of behaviour consistent with our previous knowledge of this area. We have shown in the static case that given a set goal and a set starting point, users tend to adopt similar strategies and patterns in order to reach their goal. In the dynamic scenario we do not get obvious spatial patterns as the trajectories of users are tightly coupled, leading to interactivity-based patterns of temporal behaviour.

Future work will involve collection of data of further dynamic scenarios in order to build on our knowledge of current patterns of user behaviour in such environments. We also hope to exploit the data gathered from both tests to help in the development of techniques that help reduce network traffic, and also compensate for the effects of latency.

ACKNOWLEDGEMENT

This work was funded by Enterprise Ireland Basic Research Grant SC/2002/129/.

REFERENCES

[1] E. F. Churchill, D. N. Snowdon, and A. J. Munro, *Collaborative Virtual Environments. Digital Places and Spaces for Interaction*: Springer, 2001.

[2] S. Cheshire, "It's the latency, Stupid," in *http://rescomp.standford.edu/~cheshire/rants/Latency.html*, 1996.

[3] S. K. Singhal and M. Zyda, *Networked Virtual Environments*. New York: ACM Press, 1999.

[4] J. D. Delaney, T. Ward, and S. Mcloone, "On Network Latency In Distributed Interactive Applications," presented at National University of Ireland Maynooth Postgraduate Colloquium March 28, Maynooth, Ireland, 2003.

[5] D. Delaney, T. Ward, and S. Mc Loone, "Reducing Update Packets in Distributed Interactive Applications using a Hybrid Model," presented at 16th International Conference on Parallel and Distributed Computing Systems, August 13-15, Reno, USA, 2003.

[6] Richard C. Waters, David B. Anderson, John W. Barrus, David C. Brogan, Michael A. Casey, Stephan G. McKeown, Tohei Nitta, Ilene B. Sterns, and W. S. Yerazunis, "Diamond Park and Spline: A Social Virtual Reality System with 3D Animation, Spoken Interaction, and Runtime Modifiability," in *Presence*, vol. 6, 1996, pp. 461-480.

[7] E. Frécon and M. Stenius, "Dive: A Scalable Network Architecture for Distributed Virtual Environments," in *Distributed systems Engineering Journal*, vol. 5, 1998, pp. 91-100.

[8] Epic Games, in *www.epicgames.com*

[9] Sierra Games, www.sierra.com.

[10] GarageGames Staff, "Torque License FAQ," in *http://www.garagegames.com/index.php?sec=mg&mod=resource&page=category&qid=122*, 1999.

[11] ID Software, www.idsoftware.com