BEHAVIORAL ASSUMPTION-BASED PREDICTION FOR HIGH-LATENCY HIDING IN MOBILE GAMES

Giliam J.P. de Carpentier and Rafael Bidarra

Computer Graphics and CAD/CAM Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft, The Netherlands

(G.J.P.deCarpentier/R.Bidarra)@ewi.tudelft.nl

http://graphics.tudelft.nl

KEYWORDS

High-latency hiding, Dead Reckoning, Prediction algorithms, Mobile games

ABSTRACT

While the popularity of mobile games increases with advances in processing power of mobile devices, multi-player games in fast-paced genres using the communication capabilities of these mobile devices are still rare, mostly because of the significant round trip times of the current mobile networks. Effective latency hiding techniques exist for LAN and even for WAN connection speeds, but these techniques fail in the domain of mobile networks, as the latencies can be up to 2 magnitudes larger than a typical WAN configuration. This paper introduces a new high-latency hiding paradigm based on behavioral assumptions, designed to be suitable for racing titles. Although these assumptions limit the accuracy of predicting unexpected behavior, they allow for significant improvements in hiding network latency, making it possible to create a real-time multi-player race game using today's GPRS network.

INTRODUCTION

The widely-spread support for J2ME, Brew and other mobile languages created and adopted by phone manufacturers opened up the platform to third party software developers. This allows the creation of ever more complex games as the hardware capabilities and speed continue to increase. Because the hardware inherently supports wireless communications, games could make use of these by including multi-player capabilities [1]. Depending on their genre, some games pose more severe requirements than others on both the connection speed and the round trip time. For example, chess and other multi-player puzzle games are typically less demanding on network capabilities, posing no stringent requirements regarding round trip times. On the other hand, first-person shooters and racing games require more bandwidth and lower round trip times for a prompt feedback and, ultimately, for the sake of the game-play. Sadly, although there is some human tolerance for delay, the average round trip times in current GPRS mobile networks is far above this tolerance level.

Many multi-player PC/console games, played across moderately fast networks, are able to use common prediction techniques for anticipating the location of other players. However, when using high-latency mobile phone networks, more advanced techniques are required, in order to hide the effect of such high round trip times. This paper presents several improvements on the Dead Reckoning technique [2], culminating on so-called *3-way interpolated Dead Reckoning*, which copes with high latencies by sensibly simplifying the domain using behavioral assumptions, effectively limiting the degrees of freedom where this is less critical.

PROBLEM DESCRIPTION

Among all game genres, race games are a very typical and popular genre. When multi-player races are played using PCs or game consoles across a local-area network, the high processing power and the network speed available make it possible for all player's clients to quite accurately predict and visualize the respective cars on their displays. In such a setting, uncertainties in the order of a tenth of a second are neither critical nor very perceptible when implemented using common prediction techniques.

The main reason that makes such prediction techniques impractical in a multi-player race through a mobile network is the round-trip time, i.e. the time between the moment one player, say, turns left and the moment the other players receive that move through the network, which is in the range 2-3 seconds (at least, in the current Dutch GPRS network). In practical terms, this means that during a racing game, by the time a player receives a sent location of some competitor, the latter is already considerably further on. The effects of latency on a car race game played through broadband Internet or through a mobile network can better be expressed by a position uncertainty of one car length or of one football field, respectively. As it is the prediction's task to predict the position within this uncertainty, the latter situation is simply unacceptable as inevitable prediction errors become too large to mask in a visually plausible way using more standard prediction algorithms [3].

Standard Dead Reckoning prediction implementations, for example, consist of extrapolating the course of a competitor over a period of time based on its last known 2D or 3D position and speed. Applied to a mobile car race setting with the high latencies mentioned above, this delivers rather poor results and output: cars overshoot at track curves and, upon arrival of new position and speed data, the prediction algorithm needs to compensate for huge errors, resulting in rather unnatural behavior. As a consequence of this time scale, it is of little use attempting to apply any jump masking or filtering algorithm to standard Dead Reckoning.

BEHAVIORAL ASSUMPTIONS

In order to improve the prediction performance and its adequacy to the mobile race setting, a number of behavioral assumptions can be made. The most important assumption is that the main goal of each participant in a typical race is to reach the finish in the shortest time possible. Although this will not be true for every player (e.g. some players might go for a lot of upgrades, or perhaps enjoy making nice skid marks on the track, if the game allows this), it does model an average player for typical race games.

As it is best to follow the ideal track line for the best finish time, a good prediction, to appear convincing and natural, should assume that each simulated car will at least attempt taking curves following the best (or ideal) trajectory. If afterwards (i.e. upon receiving new data) this assumption shows to have been too optimistic, the next estimate should be adjusted accordingly, in order to compensate for the divergence.

But compensating for this divergence by updating the prediction positions with the newly received data can be in itself another difficulty. Predicting positions of other cars in the race becomes much more natural and plausible if the corrections required are carried out in a smooth manner. This often requires the prediction algorithm to find a compromise between correctness and plausibility. Every new corrective prediction should be gradually applied, avoiding abrupt changes in trajectory and/or speed, possibly at the cost of a slightly deferred effect.

A final assumption is a direct consequence of the time scale at hand: there is no point in attempting to simulate or detect collisions (and similar unpredictable fast-paced changes), because what both 'colliding drivers' see on their displays is only an approximated version of the reality. Therefore, if one player collides with another player, the latter player would not necessarily collide with the first player too at that point in time as these local realities do not always coincide exactly. One could use an arbitrating server to decide whether or not these players did or did not collide; but then again, using today's mobile networks, this information would not arrive at the players' clients before about 2-3 seconds, i.e. long after the action could achieve any sensible visual feedback.

IMPROVED PREDICTION TECHNIQUES

Steering and curve behavior

As highlighted above, standard Dead Reckoning fails to properly take curves on the track, as it only acts after a large error is detected and this information has arrived at other clients. An improvement to this consists of projecting the predicted position onto the center line of the race track. This has a positive visual effect on the global race behavior, as the car keeps on track when approaching a curve. Such steering, however, bears a somewhat monotonous and unnatural appearance. A much more convincing result is achieved by projecting the predicted position not onto the track center line but onto a varied approximation of the ideal driving line.

In addition, the direction of the simulated car can be made to steer when approaching a curve. For this, we estimate a projected position in the near future, in order to query the track's ideal line direction ahead of the current position. This direction, together with the current orientation of the simulated car, is processed by a proportional-integral-derivative (PID) controller [4]. As a result, the car exhibits a more realistic steering behavior, instead of strictly following an artificially rigid approach to curves.

Predicting position and speed

A possible implementation of standard Dead Reckoning would consist of interpolating from the current predicted position to a new prediction based on more recent received position data by means of a spline curve. This curve would interpolate over time to a predicted position in the near future, using this time to mask prediction errors. Combining this correction with other factors, like (over-)steering effects to follow the track line, could prove difficult to carry out correctly, while irregular update arrival periods only adds to the problem. A second drawback of the spline fitting procedure is its computational burden on the (limited) mobile platforms. Depending on the complexity of the (pseudo-) physics model used, the algorithms described below can outperform the spline fitting procedure while eliminating much of the problems with combining the prediction with other effects.

Two-way interpolated Dead Reckoning can bring up an improvement in prediction quality. It consists of running two simulations in parallel, computing the actual predicted position by cross-fading between the results of two simulations: one simulation, (a), based on the most recent data received (instant t_n), and the other, simulation, (b), based on the data received previously to the most recent (instant t_{n-1}). The data received and processed by these simulations consists of the last-known position and speed of the car that needs to be predicted and is converted (i.e. projected) to 1D variables used as parameters to a parametric representation of the track's ideal racing line. The actual predicted position output linearly combines the results of the two simulations by cross-fading from the older simulation (b) to the newest simulation (a) over a fixed period of time, typically the average of the update frequency. This parametric position is mapped back to a 2D representation using again the parametric representation of the racing line. Initially, the weight of simulation (b) is 100% and the weight of simulation (a) 0%; throughout the period, these weights are progressively reversed, so that at the end of the period, simulation (a) predominates.

As soon as new data is received, no matter if that happens before or after the weight of simulation (a) has reached 100%, simulation (b) is replaced by a weighed average between the previous simulation (b) and simulation (a) using their current relative weight ratio. Then, simulation (a) is initialized using the new data and resets the weight for simulation (a) to 0% and for (b) to 100%. As a result of this mix between old and new data, abrupt changes are avoided in the predicted trajectory and speed. The main drawback of this approach is that the linear combination of the two simulation outcomes can be visually perceptible, turning out to be somewhat unnatural.

The best prediction results were achieved by using a *three-way interpolated Dead Reckoning* technique, which extends the two-way technique described above in two aspects. First, it performs three simultaneous simulations, one with the most recent data (instant t_n), and two others based on the data received previously (on instants t_{n-1} and t_{n-2}). In this way, the time span of influence of past received data on the prediction is doubled, which

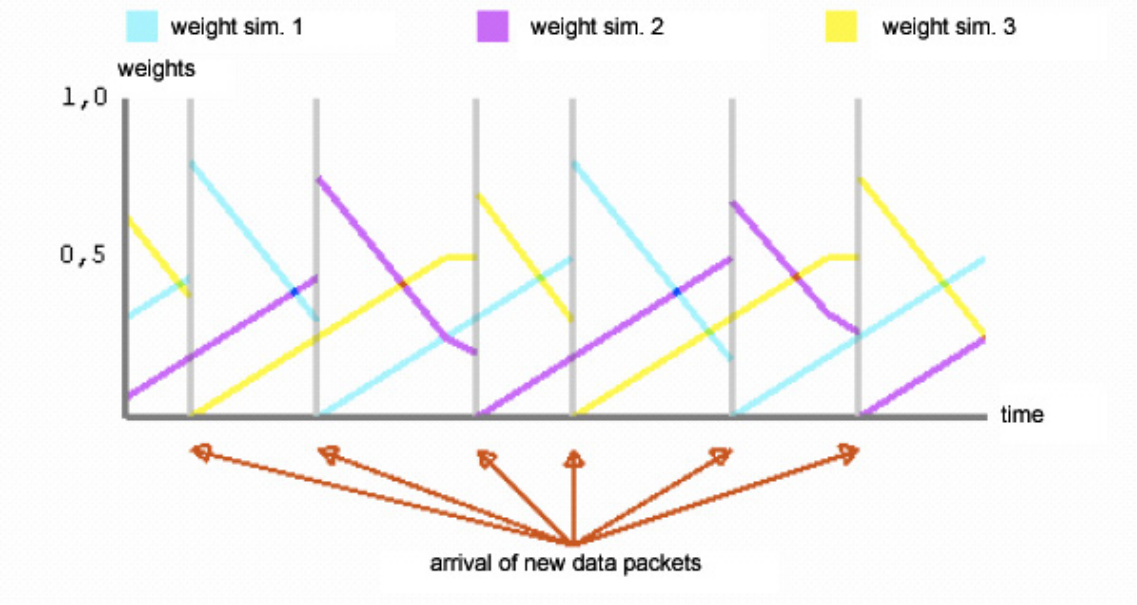

Figure 1. Combined simulation weights for three-way interpolated Dead Reckoning

has a rather positive effect on the smoothness of the simulation process.

The weights of the three simulated courses still sum up to 100% at all times, wherein the weight of the oldest simulation is linearly decreased over time (i.e. it fades out), and the weights of the other two simulations, started on successive instants, increase (with half this rate); see Figure 1. Rotating the role of these three simulations is done upon arrival of new data (at instant t_n), such that the oldest simulation (t_{n-3}) of the three is always merged by a linear combination (i.e. weighed average) with the second-oldest simulation (t_{n-2}) using their relative weights, the result becoming the new oldest simulation (t_{n-2}). This simulation gets a new weight that is the sum of the two weights that this simulation is combined from. The previously newest simulation (t_{n-1}) and its weight remain unchanged, and it becomes the second-newest simulation in the new situation. The previously oldest simulation of the three (t_{n-3}) is reinitialized with the new data to become newest simulation (t_n) with an assigned weight of 0%. One last restriction is enforced on the newest and second newest simulation by disallowing a weight contribution of more than 50% each. Although still linear, using 3 simulations somewhat masks the linearity, making the effects much less noticeable. Also, prediction errors are resolved much smoother as the newest data is less dominating. This comes at a cost of slightly larger prediction errors when compared to more direct two-way algorithms, but allows for a much more natural car behavior, as each of the simulations is still based on car (pseudo-)physics, while it is between these separate simulations that the interpolation is performed.

A second improvement that can be combined with either of the other proposed techniques consists of making use of the player's divergence from projection on the ideal racing line. This distance, between the actual last known position and its projection on the ideal race line, can be used to estimate the extra time required to drive back to the ideal line, adding a small time penalty to the appropriate simulation by temporarily slowing it down. This allows handling situations more accurately where a competitor (either willingly or unwillingly) actually diverts from the assumed route, e.g. going off the road.

RESULTS

To develop and test the above ideas, a testbed application was implemented, supporting the comparison of different prediction algorithms in real-time. Simulating a simple racing game, the tester controlled one black 'car'. All other cars in the simulation used different prediction algorithms and were fed identical positional data of the tester's car, after this data had been delayed by a simulated mobile network with limited bandwidth, large latencies and typical rates for TCP/IP resends after packet-loss. Each non-black car implemented a different algorithm for predicting the current position and speed of the tester's car, therefore

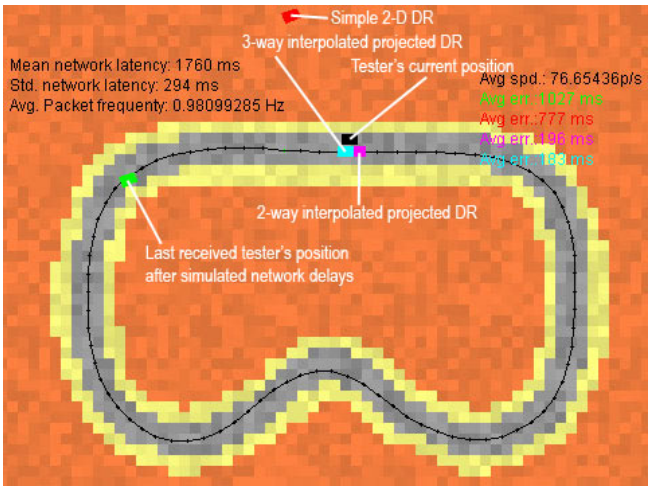

Figure 2. Testbed comparison of different prediction algorithms in on-track conditions

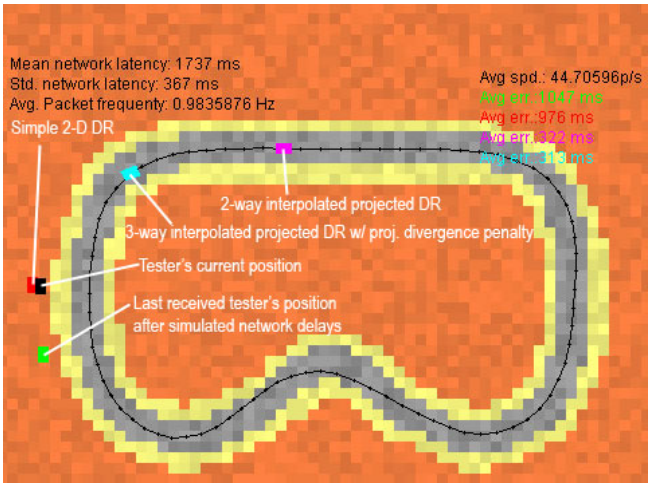

Figure 3. Testbed comparison of different prediction algorithms in off-track conditions

allowing their results to be compared. Typical results are shown in Figures 2 and 3, where the red car represents the use of a standard Dead Reckoning (2-D DR) algorithm, and the magenta and cyan car represent the proposed projected two-way and three-way algorithms, respectively. The black closed loop represents an ideal driving line used for projection. In Figure 2, the tester's objective was to race as fast as possible, a typical race game goal. Here the standard 2-D DR algorithm produces poor results compared to the proposed two-way and three-way algorithms, as it drove the simulated car mostly off-track. The average error for the two-way algorithm was slightly smaller than that for the three-way algorithm. However, the three-way algorithm produced smoother error correction if the tester behaved unexpectedly.

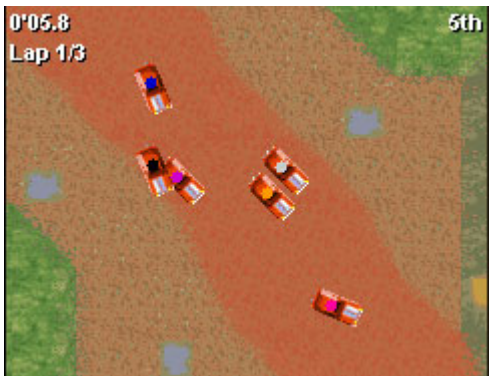

Figure 4. Scene of the implemented multi-player mobile racing game

For Figure 3, the tester deliberately drove the car off track, consequently slowing it down. In these condition, the 2-D DR algorithm still produced a poor result (Although in this particular image the red car is quite near the black tester's car). However, the proposed algorithms did not improve the prediction results of the magenta and cyan car, as the assumptions were not satisfied. Note that both the two-way and three-way algorithms overshoot the tester's position. However, the three-way algorithm also implemented a speed penalty for divergence from the ideal racing line (the last proposed improvement). Reacting to the green car (i.e. the last known position of the tester's car as known to the prediction cars) being off-track, the three-way algorithm improves the accuracy of the prediction, as expected, when compared to an algorithm without this improvement (in this case, the two-way magenta car).

CONCLUSIONS

Conventional latency hiding techniques like Dead Reckoning are not directly applicable to current multi-player mobile games, due to the high latency of the GPRS network. A new prediction technique has been presented that copes with high latencies by simplifying the domain using behavioral assumptions. This technique has been implemented for the domain of multi-player mobile race games, and tested against common alternative techniques in this domain. The new technique has shown to perform rather satisfactorily, outperforming the competing alternatives whenever typical behavioral assumptions can be made about the specific domain. So far, this technique has been incorporated into one commercial multi-player race

title for the mobile platform called Razor, by Ex Machina, running on the GPRS network; see Figure 4 for a scene taken during its development.

REFERENCES

[1] Hannay, E. "High Latency Mobile Multiplayer Gaming". http://www.lancs.ac.uk/ug/hannay/proposal.pdf

[2] Laramée, F.D. "Dead Reckoning in Sports and Strategy Games". In: AI Game Programming Wisdom 2, Charles River Media, p. 499-504

[3] Pantel, L and Wolf, L.C. "On the Suitability of Dead Reckoning Schemes for Games". http://www.ibr.cs.tu-bs.de/events/netgames2002/presentations/wolf.pdf

[4] Forrester, E. "Intelligent Steering Using PID Controllers'. In: AI Game Programming Wisdom 2, Charles River Media, p. 171-178

BIOGRAPHY

Giliam J.P. de Carpentier is a MSc computer science student at the Faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology, The Netherlands. He worked on this project within his BSc graduation, at the Computer Graphics and CAD/CAM Group of Delft University of Technology.

Rafael Bidarra is assistant professor Geometric Modelling at the Faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology, The Netherlands. He graduated in electronics engineering at the University of Coimbra, Portugal, in 1987, and received his PhD in computer science from Delft University of Technology in 1999. He teaches several courses on computer games within the CS programme 'Media and Knowledge Engineering', and leads the research work on computer games at the Computer Graphics and CAD/CAM Group. His current research interests in this area include procedural and parametric modelling, and advanced techniques for animation and path finding. He has published many papers in international journals, books and conference proceedings, and has served as member of several program committees.