On the Suitability of Dead Reckoning Schemes for Games

Lothar Pantel
Pantronics
lothar.pantel@pantronics.com

Lars C. Wolf
Technical University Braunschweig
Muehlenpfordtstraße 23
38106 Braunschweig, Germany
wolf@ibr.cs.tu-bs.de

ABSTRACT

A major problem of network-based multiplayer games (and other distributed virtual environments) is caused by the network transmission delay. This delay leads to inconsistency and other problems. Dead reckoning is often used to reduce the effects of network induced delays and losses by applying prediction means. The quality of the prediction and, hence, the consistency of the distributed game, depends on the difference between the real and the predicted position of some objects. In this paper we discuss prediction methods for games and study their usefulness in regard to different game types (genre) like Sport, 3D-Action and racing games. Using implementations, the methods are evaluated experimentally.

Keywords

Networked Computer Games, Prediction Schemes

1 INTRODUCTION

A major problem of networked multiplayer games is caused by the network transmission delay. This means that it takes a while until the information, e.g., about the new position of the opponents objects, reaches the receivers. While some technical means such as network-level quality of service mechanisms, e.g., DiffServ, can reduce or at least bound this delay, some delay will always exist, for instance about 0.1s for the propagation of light between Europe and Australia. This delay causes several difficulties and leads to paradoxical situations. As an example, consider a racing game with two players (Figure 1). The two cars are going head-to-head while passing the finishing line. Caused by the transmission delay, it takes a while until the position of the counter player reaches the local player. Therefore, the position of the opponent's car is out-of-date and somewhat behind. Thus, both players believe that they have the lead and are the winner of the game. The endsystems have no better information and will declare their local player as the winner.

This example makes clear that it is important to provide for a consistent state as far as possible. Different approaches have been developed such as:

- local presentation delay
- dead reckoning

In the first approach, the processing and presentation of events from the remote and the local system are synchronized. This requires that also local events are delayed. In [3] we investigated the impact such delay can have on the performance of multiplayer games.

With dead reckoning, the next event such as a position update is predicted. This can reduce the impact of delay and of loss of events also. However, the accuracy of the prediction is critical. In this paper we study several prediction schemes and evaluate their suitability for games by experiments.

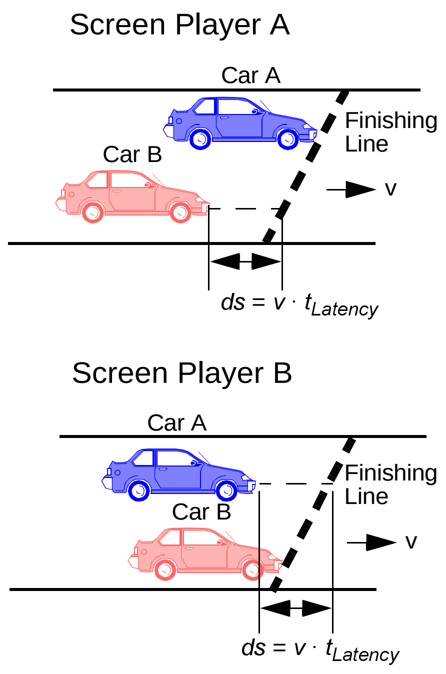


Figure 1: Example for Delay-Induced Inconsistency.

Work on dead reckoning schemes has been performed by several other groups already before. However, their work is directed towards the use of dead reckoning in distributed interactive simulation, virtual reality, and military applications. As far as we know, no investigation within the games realm has been made. This is the purpose of our paper.

The outline of this paper is as follows: The next section discusses the prediction schemes used during our experiments. Section 3 describes the evaluation scenario. In Section 4, the results of our experiments are presented before we conclude the paper.

2 PREDICTION SCHEMES

Different schemes are possible for the prediction of a future event or position. We used two categories:

- prediction of a position
- prediction of an input event

The methods within the first category predict the future position of a game object based on the current and the past positions. Not only straight lines but also curvatures can be modelled.

In the second case, not the future position of the game object but the future position and setting of the controlling input device (joystick, mouse, keyboard, etc.) is predicted. Based on the used model for the behavior of the game objects (their movements, in general a physical model), the expected user input is then used to calculate the new object positions.

In principle, a further distinction would be possible, i.e., whether the prediction is done on the sender or the receiver side. Performing the prediction at the sender can have the advantages that more information is available (more parameters of the observed object such as its physical characteristics, the environment, etc. which could be transmitted to the other systems with significant effort only) and that the computation effort is needed only once (and not on *n-1* systems if a *n* player game is used). As disadvantage we notice that sender-side prediction cannot take care of delays or loss. In this paper we concentrate on receiver-side predicition.

We studied several predicition approaches as discussed in the following.

2.1 Position Prediction Schemes

Prediction Scheme 1: Prediction of the position by assuming constant, component-wise velocity $\vec{v}_{object} = const$

Prediction Scheme 2: Prediction of the velocity by assuming constant, component-wise acceleration $\vec{a}_{object} = const$

Prediction Scheme 3: The same as scheme 2 but with a component-wise prediction using the Lagrange polynom (with a degree of 2):

$$p_i = \text{object positions}$$

$$t_i = \text{time index for event}$$

$$t = \text{to be predicted time index}$$

$$p(t) = p_0\frac{(t-t_1)(t-t_2)}{(t_0-t_1)(t_0-t_2)} + p_1\frac{(t-t_0)(t-t_2)}{(t_1-t_0)(t_1-t_2)} + p_2\frac{(t-t_0)(t-t_1)}{(t_2-t_0)(t_2-t_1)}$$

The first two schemes 1 and 2 have been derived from physical properties. We used the Lagrange polynom since it is commonly used for interpolation/extrapolation in computer graphics. Yet, schemes 2 and 3 are mathematically equivalent and, hence, are combined in the evaluations discussed below.

Prediction scheme 4: Prediction of the trajectory by assuming constant, component-wise velocity $\vec{v}_{object} = const$

2.2 Input Prediction Schemes

Prediction Scheme 5: Prediction of the input by assuming constant control device (e.g., joystick) position $\vec{s}_{controldevice} = const$

Prediction Scheme 6: Prediction of the input by assuming constant control device (e.g., joystick) velocity $\vec{v}_{controldevice} = const$

Prediction Scheme 7: Prediction of the input by using the Lagrange polynom and assuming constant control device (e.g., joystick) acceleration $\vec{a}_{controldevice} = const$

2.3 No Prediction

For comparison purposes, we introduce an additional 'scheme' where no specific but the trivial prediction 'expected position' = 'last known position' is applied. This scheme is used to put the gain of the other schemes into a useful relation.

Prediction Scheme 8: No further prediction:
expected position = current position

3 EVALUATION SCENARIO

An evaluation of prediction schemes can only be useful, if the studied game types and scenarios are chosen properly and appropriate. For this, we considered games which are

- multiplayer-capable (at least in principle), and
- controllable via proportional input devices (like joystick or mouse) and where the direction and velocity of the game objects is directly controlled by the user

The prediction schemes have been evaluated using three different application scenarios. For this purpose, according applications (simplified games) and the prediction schemes have been implemented. The application scenarios are:

- sports-games such as soccer, hockey, tennis, basketball; the player controls an object on the field.
- 3–D action games and action adventures; controlling is done from the first-person perspective or from a 'view over the shoulder'
- simulator and racing games; these are applications and games where a physical movement model applies.

A replay function has been implemented for the used games. This allows to record and replay user input, control device operation and game object positions.

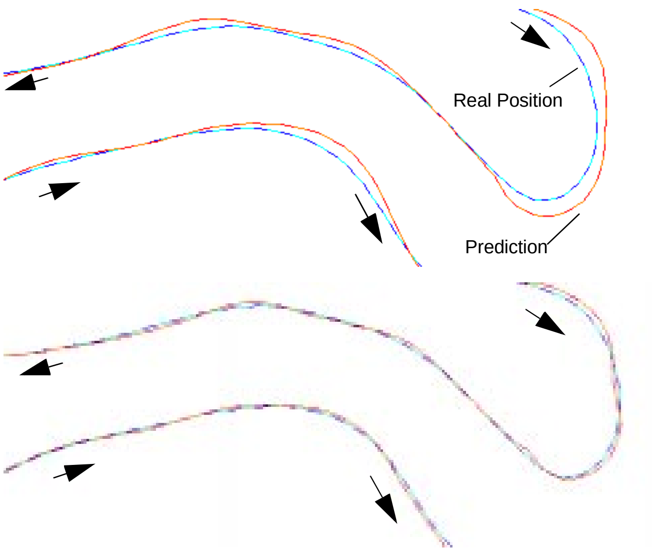


Figure 2: Deviation for racing-car game (for 200ms latency)
Top: Prediction scheme 1 (v=const);
Bottom: Prediction scheme 2/3 (a=const).

The quality of a prediction scheme is defined as the difference between the position calculated by the game engine (physical model) and the prediction. To get a general overview about the quality of a prediction scheme we determine the average deviation between the predicion and the real position of the object. Additionally, in order to get a better situation-dependent and more concrete idea of the deviations, we also implemented a viewer program. This program is used to visualize the trajectory a certain game object has driven by reading the recorded trace data and presenting it. In Figure 2 the output of the viewer program is given. It shows the predicted and the real object positions of a car-racing game for the prediction schemes 1 and 2/3 respectively if a delay of 200 ms is applied.

To study the effect of delay on the prediction schemes, all measurements have been made with simulated delay values of 100ms and 200ms.

3.1 Sports Games

For the sports-games type, we use a cursor like steering model (looking at existing games, we found this to be the common approach). We used a trace of a soccer match (recorded on video tape) as base for our study.

3.2 Action Games

Basically for all games within this category, the steering of the player's object is done in the same manner. The direction and speed values from the control device result in a target position which is used for the animation. Hence, we use one model for these two game types only. It is similar to that of steering a vehicle, e.g., moving the joy stick to the left or right causes a change of the direction; the y-axis is used for forward/backward movements.

A recorded trace of a game session is used for the examinations.

3.3 Simulator and Racing Games

The steering approach is similar to the one used above. Also here we use a trace of an example session as base for the measurements.

4 RESULTS

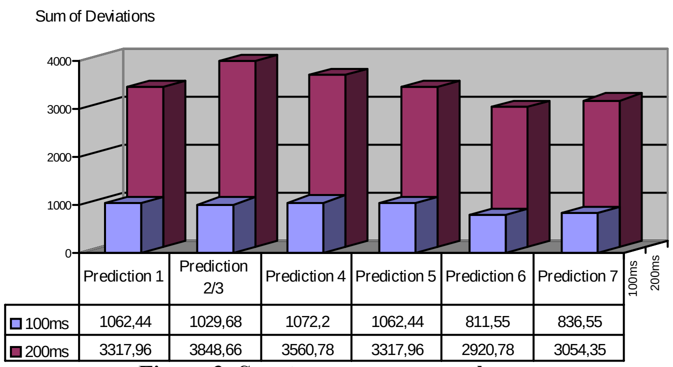
As will be discussed in more detail, our investigations show that there are measurable differences between the prediction schemes. Interestingly, the most complex schemes do not always lead to the best results. Also rather simple schemes such as prediction scheme 5 which uses the assumption of a constant control device position have been quite successful.

An important result is that the choice of the best prediction scheme depends on the used type of game, i.e., the basic control approach. While this is not necessarily much of a surprise, it is nevertheless useful to have evidence from experiments. Based on the available measurements, it is also noteworthy that the choice of the best scheme does not significantly depend on the particular player. While there are significant differences in the sum of the deviations (due to accumulations of situations which are difficult to predict), the best scheme typically leads to the lowest deviations for all playing styles. In summary, good predicition schemes (within one game genre) deliver good results for all playing styles if they are compared with other schemes.

4.1 Comparison Among Prediction Schemes

Our measurements show that the input prediction (scheme 6) with constant control device velocity leads to good results for sport-games (Figure 3). It offers the best result, i.e., the smallest average deviation, of all studied schemes, independent of the chosen delay (100ms or 200ms).

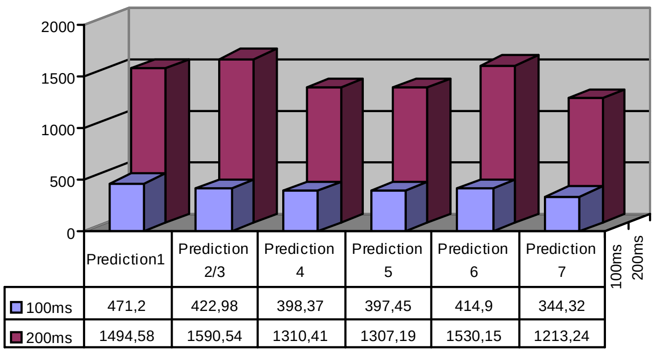
In general, the input predicition schemes (schemes 5-7) seem to be suited well for this game genre. Their results are typically better than those of the position prediction schemes (schemes 1 - 4). The remaining deviations for schemes 2 and 3 are 27% larger than those for scheme 6. The best position prediction scheme is the simple linear prediction (scheme 1).



	Prediction 1	Prediction 2/3	Prediction 4	Prediction 5	Prediction 6	Prediction 7
🟦 100ms	1062,44	1029,68	1072,2	1062,44	811,55	836,55
🟥 200ms	3317,96	3848,66	3560,78	3317,96	2920,78	3054,35

Figure 3: Sports game, average player.

The results for the 3D-action genre are shown in Figure 4. Prediction scheme 7 (the Lagrange polynom, used with a weighting factor of *w=0.3* for *t*, i.e., using *wt* instead of *t*) delivers the best results. The weighting factor *w=0.3* has been determined experimentally. To avoid this additional effort, which also depends on the chosen scenario, the simpler scheme 5 can be used which also offers good results. Further, scheme 4 (trajectory prediction) may be applied since it also leads to relatively small deviations. The reason is that the vehicle-like movements can be predicted well using this scheme. As already seen for the sports game, the constant acceleration assumption taken by schemes 2/3 does not fit well with this game type.



	Prediction1	Prediction 2/3	Prediction 4	Prediction 5	Prediction 6	Prediction 7
🟦 100ms	471,2	422,98	398,37	397,45	414,9	344,32
🟥 200ms	1494,58	1590,54	1310,41	1307,19	1530,15	1213,24

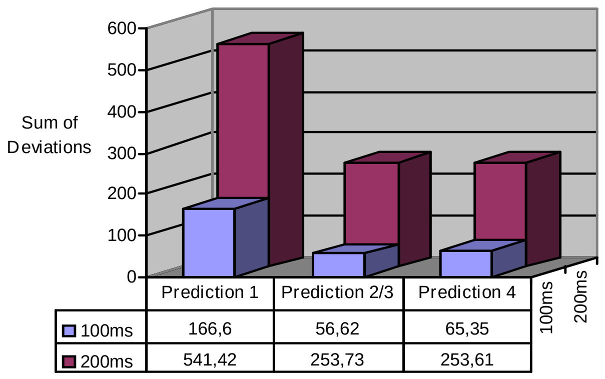
Figure 4: 3D action game, average player.

As expected, the experiments demonstrate that prediction schemes are especially well suited for the simulator genre. We used a racing-car simulator for our experiments. Only prediction schemes 1 to 4 have been evaluated because the implementation of the input prediction schemes would have required too much effort (we used a physics engine [4] which is not available in source code). The results are given in Figure 5.

For this game type and with the used prediction schemes, the deviations are in an order of magnitude smaller than without a prediction. While the average deviation was 25.6% for the sports game and 16.1% for the 3D-action genre, for the simulator game it is only 6,1% (for a delay of 200ms and using the best prediction scheme).

Moreover, compared to the other genres, we have the largest differences between the prediction schemes. The assumption 'constant acceleration' $\vec{a}_{object} = const$ used in scheme 2/3 leads to significantly better results (factor of 2 for a delay of 200 ms resp. factor of 3 for a delay of 100 ms) than the assumption 'constant velocity' $\vec{v}_{object} = const$ taken in scheme 1.

While scheme 4 leads to good results as well, the difference is not very significant and does not seem to rectify the increased implementation effort in comparisn to the simpler and similar efficient schemes 2/3.



	Prediction 1	Prediction 2/3	Prediction 4
▢ 100ms	166,6	56,62	65,35
■ 200ms	541,42	253,73	253,61

Figure 5: Racing game, average player.

4.2 Comparison With Prediction-Less Case

The previous subsection discussed the different gains achieved by the various prediction schemes and made a comparison between them. Now we look at the effect of using prediction at all, i.e., we compare the results achieved by the schemes with the case that no prediction is applied.

Figure 6 shows the remaining deviations relative to the prediction-less case for the considered game categories. It is included for a comparison of the effect of the best prediction schemes for the different game genres. As can already be seen from the previous figures, Figure 6 illustrates even more that especially racing/ simulator like games can profit from predicition schemes, i.e., the remaining deviation is lowest for this genre.

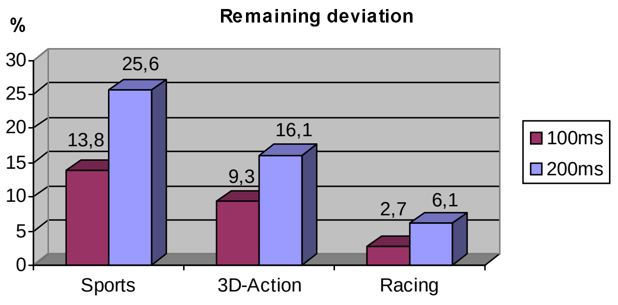


Figure 6: Overview about remaining deviations relative to prediction-less case.

Taking a closer look at Figure 6, we can notice that a vehicle-like movement model (speed and steering) is of advantage for a prediction; such a model is used in the 3D-action and racing genre. Sports games are more difficult to handle. The reason for this is that the movements taking place are not of such a continuous and smooth manner as it is the case within the other two genres. On the other hand, since the overall speed is typically lower, presentation delay schemes, e.g., [1], [2], can be applied. For faster games, like car-racing simulators, using presentation delay schemes for the improvement of the game consistency is more critical since they increase the delay which decreases the games fidelity [3].

While Figure 6 shows the remaining relative deviations, it is also interesting to look at the absolute gains. The following figures demonstrate that much worse differences occur if no prediction is applied.[1]

1. The absolute numbers are only meaningful within one type of game and cannot be compared.

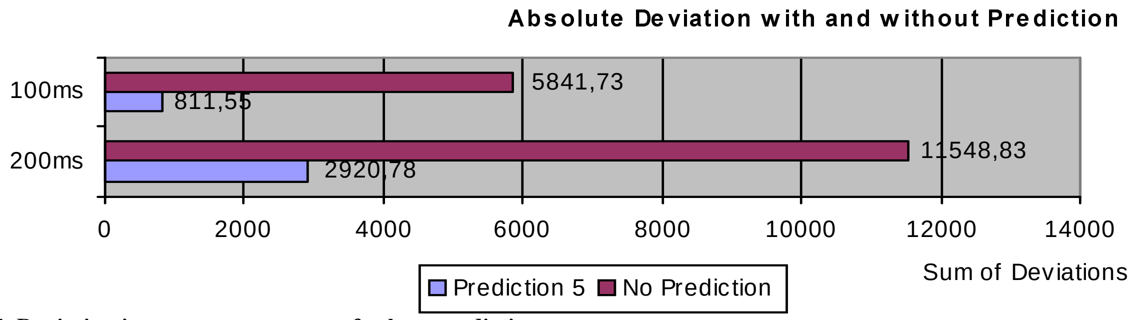


Figure 7: Deviation in sports-games genre for best prediction.

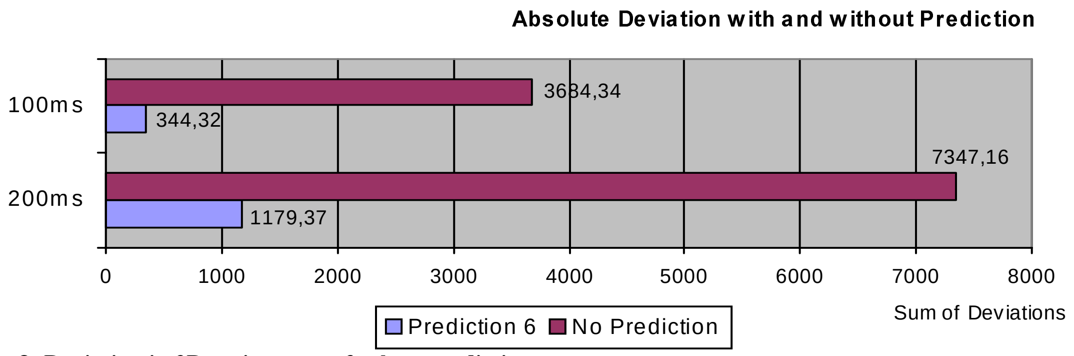


Figure 8: Deviation in 3D-action genre for best prediction.

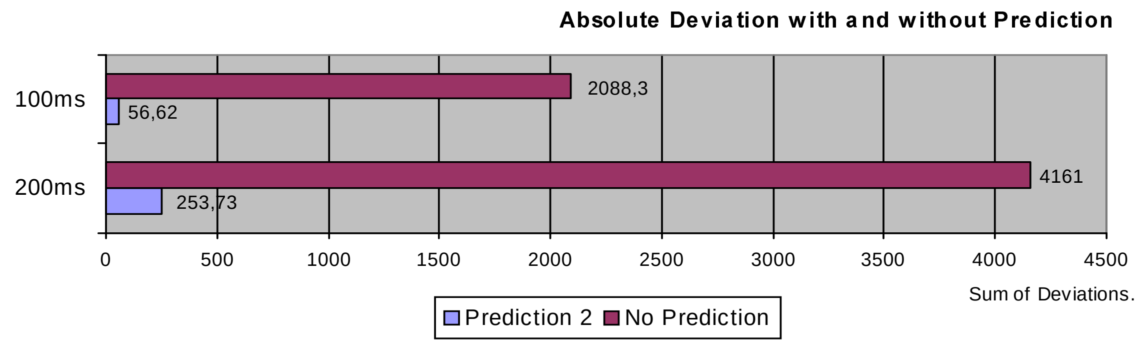


Figure 9: Deviation in simulator genre for best prediction.

For the racing-game simulator, we can give the deviation also in measurement units (since the modelling is true to scale).

Without prediction we get an average deviation per packet resp. display frame of:

- latency 100 ms: 3,5 ft = 106 cm
- latency 200 ms: 6,9 ft = 210 cm

This large deviation per packet / frame is due to the maximum speed of 80km/h in the racing game. In this particular game, RC cars have been simulated. These cars are build in a scale of 1:8, i.e., the length of a car is just 45 cm. Thus, depending on the latency, the deviation is two- to five-times the length of the car!

With prediction we get an average deviation per packet resp. display frame of:

- latency 100 ms: 0,57 ft = 17 cm
- latency 200 ms: 1,96 ft = 60 cm

For the chosen scenario, this is much more acceptable than those for the case without any prediction.

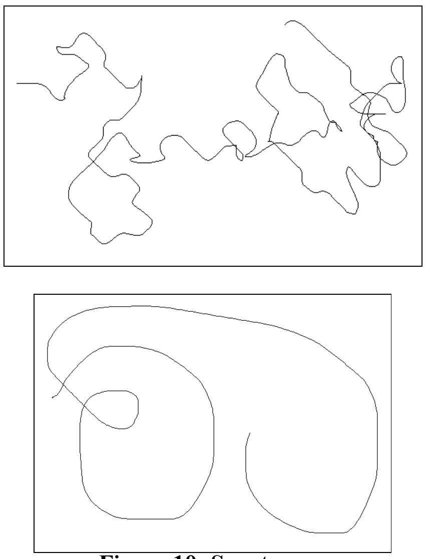


Figure 10: Sports game:
Bad and good style (with respect to prediction).

4.3 Extreme Playing Behaviors

To learn about the sensitivity of the predicition schemes to the behavior of specific players, e.g., whether they are moving the objects very smooth or very abrupt, we performed further measurements. In addition to the playing traces used above, we created unrealistic traces (with a very smooth or very hectic steering style). Figure 10 and Figure 11 illustrate examples for such playing styles.

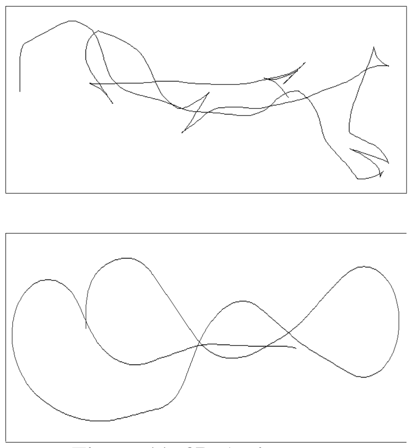
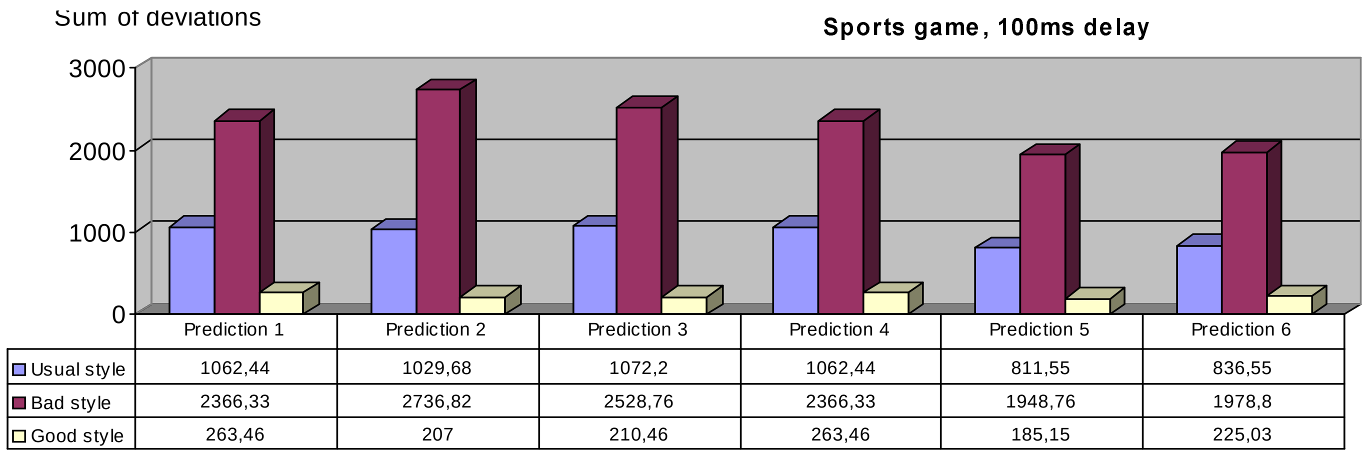


Figure 11: 3D-Action game:
Bad and good style (with respect to prediction).

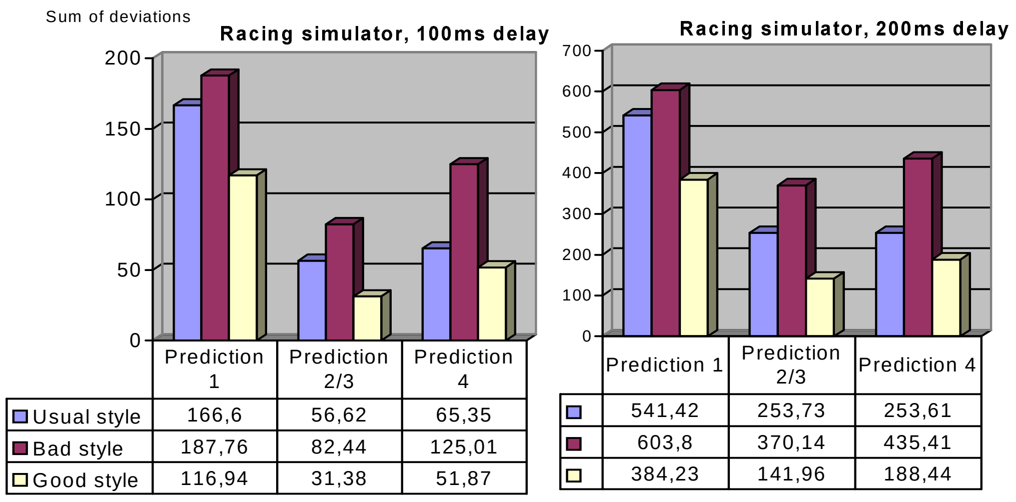
It can be seen that the playing style has a significant impact on the resulting deviation. Reasons are that hectic movements cannot be predicted well and for smooth steering operations, the velocity changes only gradually.

In Figure 12 the results for the sports game using different steering styles are shown. While the absolute values are different, the overall order among the prediction schemes does not change from style to style.



	Prediction 1	Prediction 2	Prediction 3	Prediction 4	Prediction 5	Prediction 6
🟦 Usual style	1062,44	1029,68	1072,2	1062,44	811,55	836,55
🟥 Bad style	2366,33	2736,82	2528,76	2366,33	1948,76	1978,8
🟨 Good style	263,46	207	210,46	263,46	185,15	225,03

Figure 12: Sports games: Results of prediction schemes for extreme playing styles.



	Prediction 1	Prediction 2/3	Prediction 4
🟦 Usual style	166,6	56,62	65,35
🟥 Bad style	187,76	82,44	125,01
🟨 Good style	116,94	31,38	51,87

	Prediction 1	Prediction 2/3	Prediction 4
🟦	541,42	253,73	253,61
🟥	603,8	370,14	435,41
🟨	384,23	141,96	188,44

Figure 13: Racing simulator: Results of prediction schemes for extreme playing styles.

Finally, Figure 13 gives the results for the racing game (results for the action genre are omitted, they are similar to those for the racing simulator). The differences between the playing styles are not as significant as for the sports games. This is caused by the general characteristics of these types of games. Due to the physical properties of the simulated objects, movements cannot be as abrupt as they can be for the previous game genre.

5 CONCLUSIONS

The experiments show that prediction schemes can be useful for networked games. They cannot reduce the latency directly, but they can be used to reduce its impact.

While presentation delay schemes can provide for an improved consistency of a networked game, they introduce additional delay. As we have shown in [3], a delayed presentation within a high-speed game such as a racing-car simulator is not acceptable above a certain level. Prediction schemes can be of value in that area.

References

[1] L. Gautier and C. Diot: "Design and Evaluation of MiMaze, a Multiplayer Game on the Internet", Proc. IEEE Multimedia (ICMCS'98), Austin, TX, USA, 1998, pp. 233-236.

[2] Martin Mauve: "Consistency in Replicated Continuous Interactive Media", Proc. of the ACM Conference on Computer Supported Cooperative Work (CSCW) 2000, Philadelphia, PA, USA, 2000, pp. 181-190.

[3] Lothar Pantel, Lars Wolf: "On the Impact of Delay on Real-Time Multiplayer Games", submitted for publication.

[4] J. Todd Wasson, www.PerformanceSimulations.com.