Level of Detail Based Behavior Control for Virtual Characters

Felix Kistler, Michael Wißner, and Elisabeth André

Augsburg University, Multimedia Concepts and their Applications,
D-86159 Augsburg, Germany
{kistler,wissner,andre}@informatik.uni-augsburg.de
http://mm-werkstatt.informatik.uni-augsburg.de/

Abstract. We take the idea of Level of Detail (LOD) from its traditional use in computer graphics and apply it to the behavior of virtual characters. We describe how our approach handles LOD determination and how we used it to reduce the simulation quality of multiple aspects of the characters' behavior in an existing application.

1 Introduction

Traditionally, Level of Detail (LOD) is a concept found in 3D graphics programming aiming at reducing the total amount of polygons in a scene and thus increasing the overall performance. To this end, for each 3D-Object in the scene, the distance from the camera to this object is taken into account. The further away an object is from the camera, the less important it becomes for the user and the fewer polygons are drawn for it, following the rationale that the human eye will not be able to tell the difference. Different LODs can be discrete and are switched at certain distance thresholds or they can be continuous and the number of polygons is updated dynamically. Recent research extends the idea of LOD to the simulation of the behavior of virtual characters, using the terms "Simulation LOD" or "LOD AI". The general idea is to find uninteresting or unimportant characters in the scene and reduce the simulation quality of their behavior accordingly. When applying LOD to the behavior of characters in this manner, new questions arise regarding the definition and usage of the different LODs: Is distance from the camera still a viable method to determine LOD for a certain character? Or are there situations in which a character might be interesting enough to exhibit its complete behavior, regardless of the distance to the camera? Which aspects of the behavior simulation can be reduced and how?

In this work we address the above questions and present our approach of an LOD-based behavior control system. We also report how we integrated and tested our system in an existing application, the Virtual Beer Garden. The Virtual Beer Garden is a Sims-like virtual environment where characters try to satisfy certain needs by interacting with each other or smart objects in the environment.

The novelty of our approach lies in the wide range of reducible aspects of behavior and the fact that the behavior reduction is applied at execution time and not beforehand during behavior selection.

2 Related Work

Table 1 shows an overview of previous work on "Simulation LOD" or "LOD AI", as well as our approach.

As can be seen from the table, with a wide range of reducible aspects of behavior, LOD determination based on distance and visibility and a rather large number of different LODs, our work focuses on a more flexible approach which is applicable to many different applications and scenarios.

Table 1. Comparison of different LOD approaches

Authors	**LOD based on**	**Number of LODs**	**LOD applied to**	**AI behaviors**
Chenney et al. [1]	potential visibility	2	updating movement	navigation, collision avoidance
O'Sullivan et al. [2]	distance	not specified	geometry, animations, collision avoidance, gestures and facial expressions, action selection	navigation, collision avoidance, complex dialogs with other agents
Brockington [3]	distance	5	scheduling, navigation, action selection in combat	navigation, collision avoidance, complex combat interactions
Niederberger and Gross [4]	distance and visibility	21	scheduling, collision avoidance, path planning, group decisions	navigation, collision avoidance
Brom et al. [5]	simplified distance	4	action selection (with AND-OR trees), environment simplification	navigation, complex interactions with objects and other agents
Paris et al. [6]	distance	3	navigation, collision avoidance	path planning, navigation, collision avoidance
Lin and Pan [7]	distance	not specified	geometry, animations	locomotion
Osborne and Dickinson [8]	distance	not specified	navigation, flocking, group decisions	navigation
Our work	distance and visibility	10	updating movement, collision avoidance, navigation, action execution	navigation, collision avoidance, desire-based interactions with agents and smart objects, dialogs

3 LOD Based Behavior Control System

Our LOD based Behavior Control System is based on the Horde3D GameEngine [9]. The engine is component-based and due to this modular design it can be simply enhanced by new components as done in our implementation described in this section.

3.1 LOD Determination

The most important goal in determining the different levels of detail is to keep the later usage as generic as possible. Because of that the LOD should offer enough different nuances in its classification. The current implementation calculates the LOD in up to ten configurable levels according to the distance from the camera, and further adds an again configurable value, if the object is not visible from the current camera (as suggested by Niederberger and Gross).

The different LODs for an entity in the game world can be configured within the XML-based scene graph. Figure 1 shows an example XML configuration. d0-d5 configure the discrete distances for the LOD levels in the game environment. "invisibleAdd" is the value added to the LOD in case of invisibility of the object.

In the LOD calculation, all objects whose distance to the camera is lower than d0 are assigned a LOD value of 0. For all others the following applies:

$$
\begin{aligned}
LOD(x) &= d(x) + add(x) \\
&\text{where } d(x) = i \text{ for } d_i(x) \leq r(x) < d_{i+1}(x) \\
&\text{with } r(x) = \text{ "distance of x to the camera"} \\
&\text{with } d_i(x) = \text{ "discrete distance } \mathrm{d_i} \text{ as configured for x"} \\
&\text{and } add(x) = \begin{cases} invisibleAdd(x) & \text{if } x \text{ is invisible} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

Let us assume the "invisibleAdd" of an object x is set to 2 and its distance to the camera is 100, which is between d3(x) and d4(x). Then it's first assigned an LOD value of 3. If x is also invisible for the current camera, the "invisibleAdd" is added. Thus the LOD of x becomes 5 in total.

The visibility determination takes place by testing the bounding box of an object against the camera frustum. In that way nearly visible agents are also counted as visible, in opposite to Niederberger and Gross. As most of the time only few agents fall into this category, this additional classification is of little consequence, though.

If occlusion culling is activated in the application, this extra information is also taken into account for the LOD determination. As a result objects or agents occluded by others get the "invisibleAdd" as well.

```
<AILOD d0="25" d1="40" d2="60" d3="90" d4="130" d5="200" invisibleAdd="2" />
```

Fig. 1. XML configuration for the LOD determination

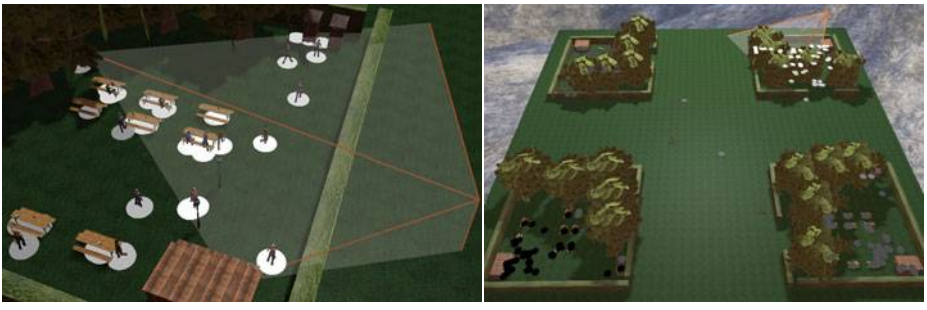

Fig. 2. LOD classification
Left: Small section of the Virtual Beer Garden environment, showing both visible and invisible agents at the same distance.
Right: Complete scene, showing many agents with LOD 7

Figure 2 shows two examples of the LOD classification. Note that the transparent, white pyramid represents the camera frustum for the LOD calculation. Also, all objects and characters for which the LOD is calculated are marked with colored circles according to their LOD value. LOD 0 is white and LODs 1-6 are incrementally darkening until black at LOD 7.

3.2 LOD Usage and Sample Application

As the Horde3D GameEngine is organized in (preferably independent) components, it seems logical to apply the Level Of Detail in each of these components separately to gain the most benefit. Moreover, each component's LOD usage can be configured individually to suit the current application's needs. However, the current LOD implementation only focuses on the AI relevant components. The following simplifications are caused by the LOD implementation so far:

- The path planning is reduced at a higher LOD level.
- The movement of the characters is simulated differently. It varies from the complete continuous movement with full animation, over continuous movement without animation, to a more and more infrequent update of the movement (so the agents move with a jerk), and ends with the direct jump to the designated destination.
- Update rates are reduced at a higher LOD.
- Repulsive forces in the governing crowd simulation are ignored from a specific LOD on. As a result, agents walk through each other and through small objects.
- Speech is only output up to a certain LOD.
- Animations are omitted at higher LODs.
- Some behavior is dropped at higher LODs.

Virtual Beer Garden. The application we applied our LOD system to is the Virtual Beer Garden [10], a typical Bavarian beer garden, with trees, benches and

tables, toilets, and a booth in which two characters sell beer and salads. Further there are waitresses which bring away empty beer steins and salad bowls. But the main characters are the patrons. They enjoy themselves by drinking beer, eating salad and chatting. They further go to the toilet, if they need. To have a more profound LOD application, we changed the environment so that it now contains for different beer gardens with a free open space in between (see figure 2 right).

The characters' movement in the application is governed by a crowd simulation which is also controlled by our LOD based system [11]. Before we describe how we applied our LOD system to the characters' AI, we will first explain it. It is based on a desire system and the use of smart objects, similar to The Sims (cf. [12], [13]).

All actions are driven by the agents' desires, which are defined by: a unique name, the current level in the range of 0 (no desire) to 100 (soon to be satisfied), an initial value (if unconfigured, initialized by a random number), an increment which continuously increases or decreases the desire, and finally a weight to prioritize desires.

To satisfy these desires, a character has to interact with the so-called smart objects, which provide information about the interaction procedure. This information normally includes exact action instructions, pre- and postconditions and additional pieces of information for the interaction, similar to the approach described by Abaci et al. [14]. The action instructions of our smart objects are divided into a sequence of preparations and a sequence of actions for the actual interaction. Both sequences are added to the characters' action queue, which they processes step by step. Possible actions include: Going to a certain position, speaking a sentence, playing an animation, picking up or setting down an object and adding a certain value to a desire. The information flow between a character and a smart object is also shown in figure 3.

In addition to the smart objects interactions, the agents can interact with each other. For example, if a character has emptied its beer stein, it can ask a waiter to refill it. If a character wants to fulfill such a desire, it first has to find

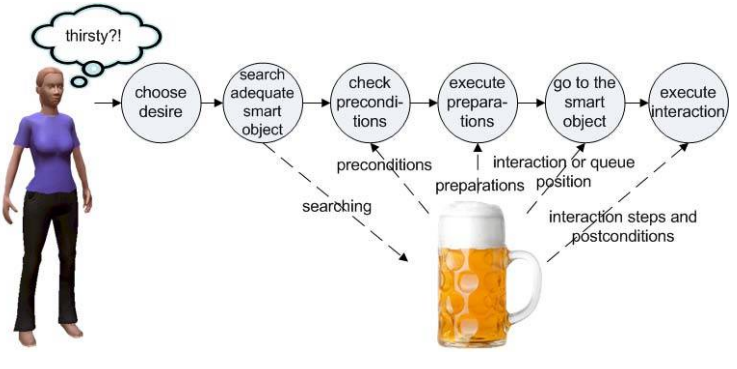

Fig. 3. Information flow between character and smart object

a partner in its search radius that feels this need, too - at least with a value of 40. If the agent has found such a partner, it will walk toward the other and waits if there is still another interaction to finish. Now, each interaction partner gets an action sequence, and they both take turns at working it off. The agent interactions in the Virtual Beer Garden mainly consist of smalltalk, but also of goal-oriented dialog, such as sales conversations between the waiter or the cook and the customer at the booth.

The characters' interaction with the smart objects is reduced by the LOD system in the following way: Based on whether a step in the interaction sequence is critical for the outcome of the interaction, the agents can skip it. To this end, the smart objects offer a maximum LOD and an alternative waiting time for each step of the interaction. At each step, the agent checks its LOD against this maximum and if it's above, the agent waits for the indicated amount of time instead of actually performing the action.

As the action selection is predetermined by the smart objects interactions, there is a high guarantee for a consistent behavior in the game progress, even if the LOD applications skips the partial steps of the interactions. This reveals an advantage over other AI techniques.

4 Conclusion and Future Work

In this work we presented our idea of an LOD based behavior control system for virtual characters based on discrete distances from an application's camera and also reported how we applied it to an existing application. We showed that many aspects of the characters' behavior can be reduced with our implementation due to its general approach. Applying the behavior reduction at the action execution level also provides us with high consistency during LOD changes.

Ideas for future work include three different kinds of improvement: First, the approach could be extended to additional behaviors, such as the gaze behavior system we described in [15] or other sensory input. Second, the approach could be made more dynamic, especially regarding the animations. As of now, they are either played or not. Here, it would be interesting to created simplified animations that could be used in between, as described by [16]. Finally, the LOD approach could also be adapted to other behavior control techniques for virtual characters, such as HTN Planners, Hierarchical Finite State Machines or Behavior Trees.

References

1. Chenney, S., Arikan, O., Forsyth, D.A.: Proxy Simulations For Efficient Dynamics. In: Proceedings of Eurographics (2001)
2. O'Sullivan, C., Cassell, J., Vilhjalmsson, H., Dingliana, J., Dobbyn, S., McNamee, B., Peters, C., Giang, T.: Levels of detail for crowds and groups. Computer Graphics Forum 21(4), 733–742 (2002)
3. Brockington, M.: Level-Of-Detail AI for a Large Role-Playing Game. In: AI Game Programming Wisdom, pp. 419–425. Charles River Media (2002)

4. Niederberger, C., Gross, M.: Level-of-detail for cognitive real-time characters. The Visual Computer: Int. Journal of Computer Graphics 21(3), 188–202 (2005)
5. Brom, C., Šerý, O., Poch, T.: Simulation Level of Detail for Virtual Humans. In: Pelachaud, C., Martin, J.-C., André, E., Chollet, G., Karpouzis, K., Pelé, D. (eds.) IVA 2007. LNCS (LNAI), vol. 4722, pp. 1–14. Springer, Heidelberg (2007)
6. Paris, S., Gerdelan, A., O'Sullivan, C.: CA-LOD: Collision Avoidance Level of Detail for Scalable, Controllable Crowds. In: Proc. of the 2nd Int. Workshop on Motion in Games, pp. 13–28. Springer, Heidelberg (2009)
7. Lin, Z., Pan, Z.: LoD-Based Locomotion Engine for Game Characters. In: Hui, K.-c., Pan, Z., Chung, R.C.-k., Wang, C.C.L., Jin, X., Göbel, S., Li, E.C.-L. (eds.) EDUTAINMENT 2007. LNCS, vol. 4469, pp. 214–224. Springer, Heidelberg (2007)
8. Osborne, D., Dickinson, P.: Improving Games AI Performance using Grouped Hierarchical Level of Detail. In: Proc. of the 36th Annual Convention of the Society for the Study of Artificial Intelligence and Simulation of Behaviour (2010)
9. Horde3D GameEngine: University of Augsburg (2010), `http://mm-werkstatt.informatik.uni-augsburg.de/projects/GameEngine/`
10. Rehm, M., Endrass, B., Wißner, M.: Integrating the User in the Social Group Dynamics of Agents. In: Workshop on Social Intelligence Design, SID (2007)
11. Wißner, M., Kistler, F., André, E.: Level of Detail based Behavior Control for Virtual Characters in a Crowd Simulation. In: CASA 2010 Workshop on Crowd Simulation (2010)
12. Champandard, A.J.: Living with The Sims' AI: 21 Tricks to Adopt for your Game (2010), `http://aigamedev.com/open/highlights/the-sims-ai/`
13. Tirrell, J.W.: Dumb People, Smart Objects: The Sims and the Distributed Self (2010), `http://www.jtirrell.com/jtirrell/dumb_people`
14. Abaci, T., Cíger, J., Thalmann, D.: Planning with smart objects. In: The 13th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 25–28 (2005)
15. Wißner, M., Bee, N., Kienberger, J., André, E.: To See and to Be Seen in the Virtual Beer Garden - A Gaze Behavior System for Intelligent Virtual Agents in a 3D Environment. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS, vol. 5803, pp. 500–507. Springer, Heidelberg (2009)
16. Giang, T., Mooney, R., Peters, C., O'Sullivan, C.: ALOHA: Adaptive Level of Detail for Human Animation: Towards a new Framework. In: EUROGRAPHICS, Short Paper Proceedings, pp. 71–77 (2000)