

Level of detail AI for virtual characters in games and simulation

Michael Wißner, Felix Kistler, Elisabeth André

Angaben zur Veröffentlichung / Publication details:

Wißner, Michael, Felix Kistler, and Elisabeth André. 2010. "Level of detail AI for virtual characters in games and simulation." In Motion in games: Third International Conference, MIG 2010, Utrecht, The Netherlands, November 14-16, 2010, edited by Ronan Boulic, Yiorgos Chrysanthou, and Taku Komura, 206-17. Berlin [u.a.]: Springer. https://doi.org/10.1007/978-3-642-16958-8_20.



Level of Detail AI for Virtual Characters in Games and Simulation

Michael Wißner, Felix Kistler, and Elisabeth André

Institute of Computer Science, Augsburg University,
86135 Augsburg, Germany
{wissner,kistler,andre}@informatik.uni-augsburg.de

Abstract. Following recent research that takes the idea of Level of Detail (LOD) from its traditional use in 3D graphics and applies it to the artificial intelligence (AI) of virtual characters, we propose our approach on such an LOD AI. We describe how our approach handles LOD classification and how we used it to simplify the simulation quality of multiple aspects of the characters' behavior in an existing application. Finally, we delineate how we evaluated our approach with both a performance and perception analysis and report our findings.

1 Introduction

In its classical meaning, Level of Detail (LOD) is a concept found in 3D games and simulations. It aims at reducing a scene's geometrical complexity and thus increasing the overall performance. To this end, each object's importance in the scene is determined, usually by measuring its distance from the camera and thus the player. The less important an object becomes for the player (i.e. the further away it is from his point of view), the fewer the number of polygons it is rendered with. This follows the rationale that at those distances, the human eye will not be able to spot any differences. Different LODs can be discrete, meaning they are switched as certain distance thresholds are crossed or they can be continuous and the number of polygons is updated dynamically. Recent research applies this traditional idea of LOD to virtual characters' artificial intelligence (AI) and the simulation of their behaviors, using the terms "Simulation LOD" or "LOD AI". Similar to the original approach, this is based on finding uninteresting or unimportant characters in the scene and reducing the simulation quality of their behavior accordingly. When building such an LOD AI for virtual characters, not all concepts can be simply transferred from the classical meaning of LOD. New questions arise, concerning how the different LODs should be defined and used: Is a character's distance from the player still a viable method to determine its LOD? Are there maybe situations in which a character might be interesting or important enough to show its complete behavior, regardless of where it is? Which aspects of the characters' behavior can be simplified and how? Will a player be able to spot these simplifications? Will the characters and their behavior still appear believable and consistent to the player?

In this work we put forward our approach to LOD AI and try to answer the above questions. We also report how we integrated and tested our system in two existing applications. The first application is Augsburg3D, a crowd simulation featuring 250 virtual characters in a reconstruction of the inner city of Augsburg. Aspects of the characters' behavior that are reduced by the LOD AI include path finding, animation and movement and the particle-based crowd simulation itself. The second application is the Virtual Beer Garden, a Sims-like simulation where characters' behavior is driven by certain needs which they try to satisfy through the interaction with other characters or smart objects. In addition to the previously mentioned crowd simulation related behaviors, this application also combines our LOD AI with these smart objects.

The novelty of our approach to a LOD AI lies in the many different types of behavior that can be reduced with it. Also, behaviors are not reduced during the behavior selection process but later, during behavior execution.

Performance tests and a user study show that the LOD-based behavior control increased the performance by a considerable amount while users barely noticed any difference in the characters' behavior.

The remainder of this paper is organized as follows: The next chapter introduces related work. Chapter 3 describes our LOD AI and how it was applied to the Augsburg3D application. Chapter 4 presents the results of the performance analysis and user perception study we conducted to evaluate our system and chapter 5 finally gives a conclusion and a short outlook on future work.

2 Related Work

Although they do not explicitly use the term "LOD" or reduce their characters' behavior simulation, the autonomous pedestrians described by Wei and Terzopoulos [1] are relevant for our work, as they display navigation and collision avoidance, interactions with other agents and the environment as well as dialogs. Patel et al. [2] describe dialog behavior for 'Middle Level of Detail Crowds', i.e. characters that are in the background but nevertheless important.

There is little previous research on the validation of crowd simulations with virtual characters through user perception: Pelechano et al. [3] describe how to validate crowds by making the user part of them and using the notion of presence. Peters and Ennis [4] showed short animations to users, asking them whether the groups in the animations were plausible.

Moving on to "Simulation LOD" or "LOD AI", Table 1 shows an overview of previous work on this topic, as well as our approach, characterized by the following criteria: LOD determination, LOD usage and actions performed by the AI.

Looking at the table, we see that our work offers a more flexible approach, with many different types behaviors involved, LOD determination based on distance and visibility and a rather large number of different LODs. Due to this flexibility, it can be used 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. [5]	potential visibility	2	updating movement	navigation, collision avoidance
O’Sullivan et al. [6]	distance	not specified	geometry, animations, collision avoidance, gestures and facial expressions, action selection	navigation, collision avoidance, complex dialogs with other agents
Brockington [7]	distance	5	scheduling, navigation, action selection in combat	navigation, collision avoidance, complex combat interactions
Niederberger and Gross [8]	distance and visibility	21	scheduling, collision avoidance, path planning, group decisions	navigation, collision avoidance, path planning
Pettré et al. [9]	distance and visibility	5	geometry, updating movement, collision avoidance, navigation	navigation, collision avoidance, path planning
Brom et al. [10]	simplified distance	4	action selection (with AND-OR trees), environment simplification	navigation, complex interactions with objects and other agents
Paris et al. [11]	distance	3	navigation, collision avoidance	path planning, navigation, collision avoidance
Lin and Pan [12]	distance	not specified	geometry, animations	locomotion
Osborne and Dickinson [13]	distance	not specified	navigation, flocking, group decisions	navigation
Our approach	distance and visibility	8	updating movement, collision avoidance, navigation, action execution	navigation, collision avoidance, desire-based interactions with agents and smart objects, dialogs

3 Level of Detail Based AI

Our LOD based AI for virtual characters is based on the Horde3D GameEngine (cf. [14]). The GameEngine is organized into different components such as a scene graph component, an animation component, a crowd simulation component, and a text to speech component. Because of this modular design, it can be simply enhanced by new components, as we did with our approach.

3.1 LOD Classification

The most important goal in classifying the different LODs is to keep their later usage as generic as possible. Because of that, the approach should offer a sufficient number of LODs. The current implementation classifies LOD in one of up to ten configurable levels. The main criterion for this classification is an entity's distance from the camera. Furthermore, if the entity is not visible, an additional and again configurable value, is added (as suggested by Niederberger and Gross). An entity's LOD configuration is placed in a sub node of its XML-entry for the GameEngine. This means that different entities can have different LOD configurations, adding to the applicability of the LOD AI to many different applications, scenarios and circumstances. As examples could be mentioned: the size and complexity of the environment, the camera perspective and its possible movements or specific restrictions given by the game logic. Figure 1 shows an example XML configuration. The attributes d0-d5 describe the discrete distances for the LOD levels in the game's environment, and "invisibleAdd" is the value added to the LOD in case the entity is not visible.

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

Fig. 1. XML configuration for the LOD determination

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

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

As an example, let us assume the following: An entity's "invisibleAdd" is set to 2 and its distance to the camera is 100, which is between d3 and d4. So it is first given an LOD value of 3. If the entity should also be invisible as far as the current camera is concerned, the "invisibleAdd" is added, making the entity's LOD a total of 5.

The visibility determination takes place by testing bounding box of each entity that is subject to the LOD AI against the camera frustum. In that way almost visible agents are also counted as visible, in opposite to Niederberger and Gross. As most of the time only few entities fall into this category, the additional effort of this calculation is of little consequence, though.

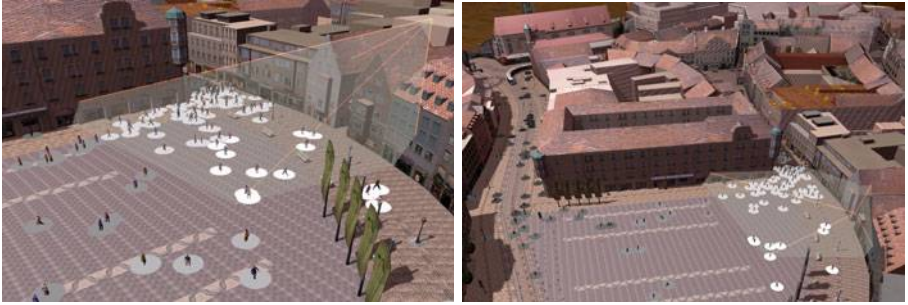


Fig. 2. LOD classification. **Left:** Small section of the Augsburg3D environment, showing both visible and invisible agents at the same distance. **Right:** Complete scene, showing many agents with LOD 7.

If the GameEngine’s occlusion culling is activated, this is also taken into account when determining an entity’s final LOD: Entities occluded by others or obstacles are considered invisible and thus receive the “invisibleAdd” as well.

Figure 2 shows two examples of the LOD classification. Note that the transparent, white pyramid represents the frustum of the camera on which the LOD classification is based. Everything outside that frustum is invisible. All objects and characters 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 Applications

As the Horde3D GameEngine is separated into independent components, it seems logical to apply the Level Of Detail AI to each of them separately. Moreover, each component’s LOD usage can be configured individually to suit the current application’s needs.

Based on this, the following simplifications are applied to the characters’ behavior so far:

- Path planning is simplified at higher LOD levels. Characters just select any route to their target, not necessarily the shortest one.
- Characters’ movement is simulated differently: It ranges from a complete and continuous movement with full animations, over continuous movement without animations, 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 desired destination.
- Repulsive forces in the later discussed crowd simulation are ignored from a certain LOD on. As a result, agents walk through each other and through small obstacles.
- Characters’ speech is only output up to a certain LOD.
- Animations are omitted at higher LODs.
- Characters lip-sync their dialogs only up to a certain LOD.
- Some parts of the characters’ interactions are omitted at higher LODs.

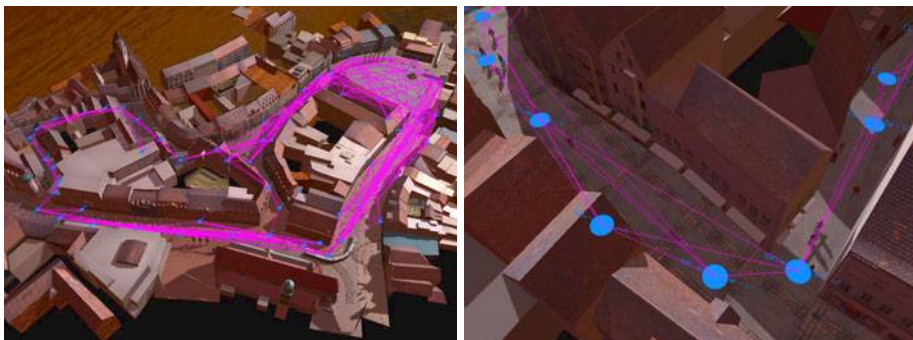


Fig. 3. Way points (blue) and the visibility graph (magenta) in the crowd simulation

Augsburg3D. Augsburg3D is a simulation of the inner city of Augsburg. The city is populated by virtual humans whose behavior is given by a crowd simulation (see [15]). This crowd simulation contains three basic elements: particles (the characters), obstacles and a net of way points. Using these way points, a visibility graph is generated, containing information about the reachability and distance among all way points. Figure 3 shows this for the whole scene (left) and in more detail (right).

Basically, the particles follow random paths between the way points. On their way, repulsive forces are applied between particles and obstacles on the one hand and between the particles themselves on the other hand. The way in these forces are applied in our crowd simulation is based on ideas by Heigeas et al. [16]. Figure 4 shows a character along with its three avoidance zones. As performance decreased with higher numbers of characters, we decided to apply level of detail to the characters' behavior. For that goal, the agents first get a LOD assigned as described in section 3.1. The discrete distances were defined as shown in Figure 1. Thus, all particles get a LOD between 0 and 5 according to the distance to the camera and a value of 2 is added in case of invisibility.

Now, two critical LOD values can be defined: One for the crowd simulation component and one for the movement animation component.

The latter is responsible for playing the characters' walk animations in an appropriate speed while they are moving. Further, it plays idle animations when the characters are standing around. If a character' LOD is above the defined critical value, walk and idle animations are simply skipped. Thus, the component adds no animation to the agent. The



Fig. 4. Three avoidance zones d1 (blue), d2 (green) and d3 (red) in the crowd simulation

critical value for the movement animation component was set to 3 for Augsburg3D. That equates to a distance of 40 at invisibility or a distance of 90 at visibility, at which the agents in Augsburg3D are small enough, that the missing animation is not too conspicuous for the user.

The crowd simulation component applies three simplifications if its critical LOD value (also set to 3 for Augsburg3D) is reached:

1. The repulsive forces are dropped, so particles walk through small obstacles like lanterns or benches and also through other particles.
2. The path calculation is reduced. In that way, the start and end point is chosen by the first found possible way point.
3. The update rate for the particle movement is reduced. Therefore, every agent stores the time past since the last update. If an update occurs, the stored time is added to the current frame time to calculate the movement. As a consequence, the particles jump over longer distances at once after several frames, instead of changing their position incrementally after every frame. As the LOD calculation is only done if the character (or camera) has changed it's position, its LOD value is also calculated less frequently (at a static camera). The update rate $f_{update}(x)$ in updates per second for particle x with LOD value $lod(x)$ is defined as following:

$$f_{update}(x) = \begin{cases} \frac{1}{t(x)} & \text{if } lod(x) > criticalLOD(x) \\ \infty & \text{otherwise} \end{cases}$$

where $t(x) = 0.55 \cdot lod'(x) - 0.5$
and $lod'(x) = lod(x) - criticalLOD(x)$

With $criticalLOD(x) = 3$ that results in the following update rates:

LOD	0	1	2	3	4	5	6	7
f_{update}	∞	∞	∞	∞	20	1,66	0.87	0.59

$f_{update}(x)$ always indicates the maximum update rate of x . Is the current frame rate below that value, the update rate is adequately smaller. Thus, ∞ is equivalent to an update in every frame. The time $t(x)$ until the next update is given by a simple linear function. The chosen gradient (0.55) and the y-intercept (0.5) ensure that, with a LOD one above the critical value, there is still a continuous movement with 20 updates per second. More complex functions did not provide an increased benefit in our experiments.

Additionally, if their LOD value is low enough, the agents also exhibit a new behavior to make the simulation more authentic: The pedestrians start simple conversations. For this matter, they search for a partner in their environment, that also has a sufficient small LOD. The two characters go toward each other, and the initiator starts a generated dialog as he reaches his partner. After they have exchanged some sentences, they both go their own ways again. To coordinate the different conversations, the particles allocate a so called interaction

slot. There is only a limited number of those slots (six in the current configuration), so that the number of parallel dialogs in the simulation is limited as well. There is further a cool down timer (currently 5 to 15 seconds) for each agent to ensure he does not start a new conversation immediately after his last one. As the LOD value is dependent on the camera distance, the conversations only occur close to the camera, the only place where the observer really notices them. The actual speech output with lip synchronization only takes place if the LOD is equal or less than one. Otherwise, the current utterance is skipped and the agent just waits two seconds. The exact duration does not matter because the conversations consist of small talk which can be easily interrupted and resumed at any time without causing visible inconsistencies in the characters' behavior.

Virtual Beer Garden. The second application with LOD usage is the Virtual Beer Garden [17]. Details on the agents' artificial intelligence and how LOD is applied to it can be found in [18].

4 Evaluation

In order to evaluate the effectiveness of our approach, we compared two versions of Augsburg3D: one including our LOD system, and one without it. In the evaluation, we addressed the following two goals:

1. For the same number of agents, the performance of the application using the LOD system should be better, i.e. the frame rate should be higher.
2. The difference in the quality of the agents' behavior between the two versions should not be recognizable by a user.

To validate goal 1, we carried out a performance analysis of the Augsburg3D application. For goal 2, we conducted a web-based perception survey of both our applications.

4.1 Performance Analysis

We created two versions of the Augsburg3D application as described above. In both versions, we populated the scene with 50, 100, 150 and finally 250 agents. Each of these four configurations was run five times, for a total of 20 runs per version. Each run lasted 45 seconds and the camera position within the application was fixed for the entire time. We recorded average frame rates for each run and averaged those for all runs of a version. For comparison purposes, we also recorded another data set based off a third version which used the LOD system and the engine's occlusion culling. Figure 5 shows the results for all three versions. The machine we used is a Intel Core 2 Quad Q9550 with 2,83 Ghz, 4 GB Ram and a Nvidia Geforce GTS 250 with 512 MB Ram.

As we can see, the two versions using the LOD system have both an increased frame rate as well as a slower decline of the frame rate for larger numbers of agents. So, goal 1 can be seen as fulfilled.

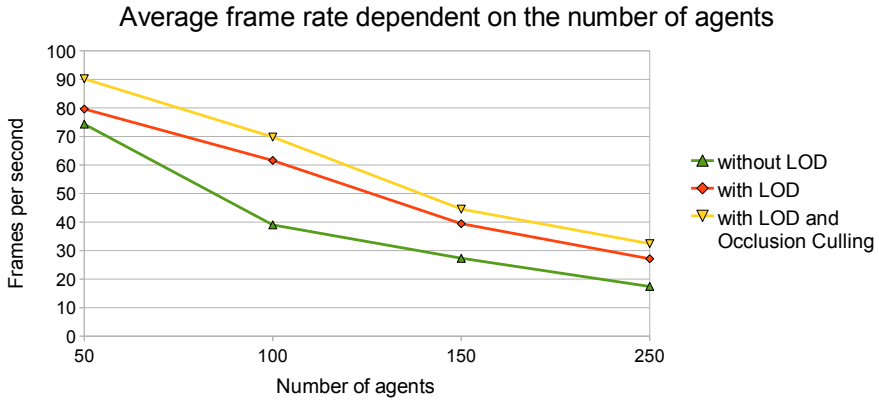


Fig. 5. Performance analysis

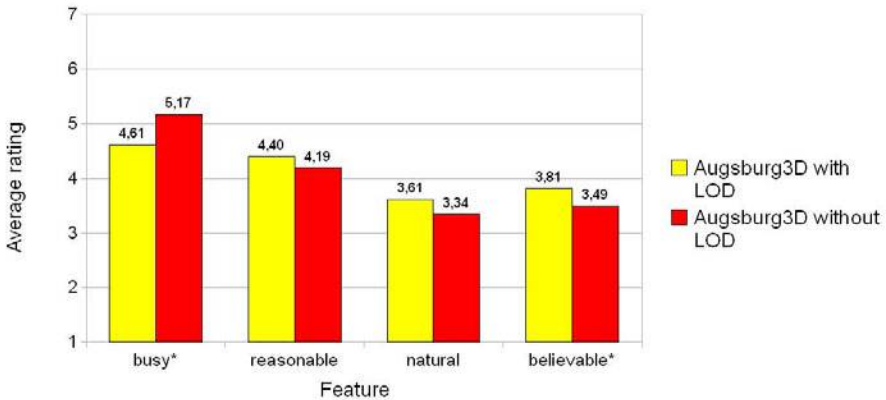


Fig. 6. Subjects' perception of Augsburg3D application, * indicates a significant difference

4.2 Perception Survey

For both the Augsburg3D as well as the Virtual Beer Garden application, we created two versions as described above. From each version we recorded a video of about 30 seconds. We created a web-based survey, asking the participants to watch the agents' behavior in the four videos and rate the following features of the scene on a 7-point Likert scale (1 to 7, 7 meaning full agreement): busy (in a positive sense, i.e. "alive"), reasonable, natural and believable. For each application, the order of the two videos was determined randomly, but the Augsburg3D videos were always shown first. Our study had 108 participants (69 male, 39 female) between ages 15 and 46. Figure 6 shows the mean values of their ratings for the Augsburg3D application.

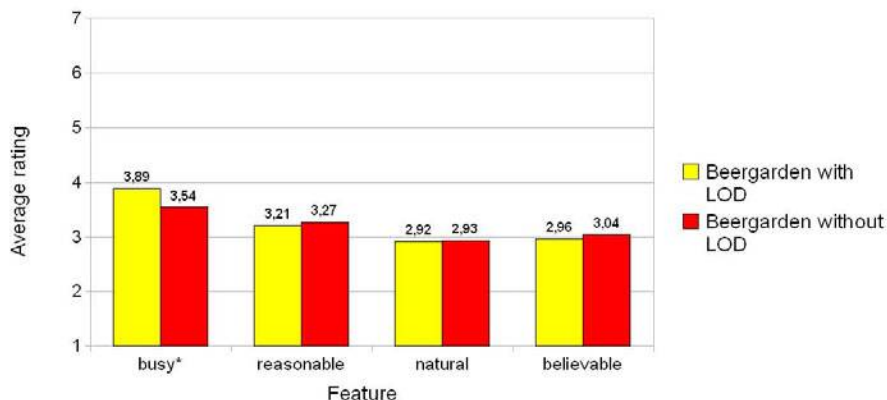


Fig. 7. Subjects’ perception of Beer Garden application, * indicates a significant difference

It can be seen that subjects thought that the “normal” version was significantly busier, with mean values of 4.61 vs. 5.17 ($t(107)=-4.599$, $p < 0.001$). However, they rated the LOD-version slightly better regarding the other features, with the difference for “believable” also being significant with mean values of 3.81 vs. 3.49 ($t(107)=2.040$, $p < 0.05$). The effect size was medium for all four features.

Figure 7 shows the mean values of the subjects’ ratings for the Virtual Beer Garden application.

Here, we get the exact opposite: Subjects rated the LOD-version busier, but thought that the normal version was better otherwise. Of these differences, only the one for “busy” was significant with mean values of 3.89 vs. 3.54 ($t(107)=3.930$, $p < 0.001$). The effect size was medium for “busy” and small for the other features.

Looking at these results with regards to goal 2, we have to state that the users noticed some differences. However, two of the three significant differences are actually in favor of our LOD system. So while there is definitely room for improvement regarding the busyness of Augsburg3D with the LOD system we see goal 2 as almost fulfilled.

5 Conclusion and Future Work

In this work we presented our idea of a Level of Detail AI for virtual characters based on discrete distances and visibility from a player’s point of view. We applied it to an existing application and showed that many aspects of the characters’ behavior can be simplified with our system due to its general approach. Applying the behavior reduction at the behavior execution level also provides us with high consistency during LOD changes.

An evaluation showed that our LOD AI considerably increased the application's performance. At the same time, users only barely observed changes in the characters' behavior (with the exception mentioned above), which suggests that our approach is viable but could also use some improvement.

Our ideas for future work include the following categories of improvement: First, the approach could be extended to additional of the characters' behaviors already present in the GameEngine, such as the gaze behavior system we described in [19]. Second, the usage of LOD in some components could be made more dynamic, especially regarding the animations: As of now, they are either fully played or not played at all. Here, it would be interesting to follow the approach described by [20] and create simplified animations that could be used in between. Finally, our LOD approach could also be adapted to other AI techniques used in games, such as HTN Planners, Hierarchical Finite State Machines or Behavior Trees.

References

1. Shao, W., Terzopoulos, D.: Autonomous Pedestrians. In: Proceedings of the 2005 ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation, SCA 2005, pp. 19–28. ACM Press, New York (2005)
2. Patel, J., Parker, R., Traum, D.: Simulation of Small Group Discussions for Middle Level of Detail Crowds. In: Army Science Conference (2004)
3. Pelechano, N., Stocker, C., Allbeck, J., Badler, N.: Being a Part of the Crowd: Towards Validating VR Crowds Using Presence. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2008, Richland, International Foundation for Autonomous Agents and Multiagent Systems, pp. 136–142 (2008)
4. Peters, C., Ennis, C.: Modeling groups of plausible virtual pedestrians. *IEEE Computer Graphics and Applications* 29(4), 54–63 (2009)
5. Chenney, S., Arikan, O., Forsyth, D.A.: Proxy Simulations For Efficient Dynamics. In: Proceedings of EUROGRAPHICS 2001 (2001)
6. 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)
7. Brockington, M.: Level-Of-Detail AI for a Large Role-Playing Game. In: *AI Game Programming Wisdom*, pp. 419–425. Charles River Media, Hingham (2002)
8. 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)
9. Pettré, J., de Heras Ciechomski, P., Maïm, J., Yersin, B., Laumond, J.P., Thalmann, D.: Real-time navigating crowds: scalable simulation and rendering. *Comput. Animat. Virtual Worlds* 17(3-4), 445–455 (2006)
10. 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.) *IWA 2007. LNCS (LNAI)*, vol. 4722, pp. 1–14. Springer, Heidelberg (2007)
11. Paris, S., Gerdelan, A., O'Sullivan, C.: CA-LOD: Collision Avoidance Level of Detail for Scalable, Controllable Crowds. In: Egges, A. (ed.) *MIG 2009. LNCS*, vol. 5884, pp. 13–28. Springer, Heidelberg (2009)

12. 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)
13. 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)
14. Horde3D GameEngine: University of Augsburg (2010), <http://mm-werkstatt.informatik.uni-augsburg.de/projects/GameEngine/>
15. Dorfmüller-Ulhaas, K., Erdmann, D., Gerl, O., Schulz, N., Wiendl, V., André, E.: An Immersive Game - Augsburg Cityrun. In: André, E., Dybkjær, L., Minker, W., Neumann, H., Weber, M. (eds.) PIT 2006. LNCS (LNAI), vol. 4021, pp. 201–204. Springer, Heidelberg (2006)
16. Heigeas, L., Luciani, A., Thollot, J., Castagné, N.: A Physically-Based Particle Model of Emergent Crowd Behaviors. In: Graphicon (2003)
17. Rehm, M., Endrass, B., Wißner, M.: Integrating the User in the Social Group Dynamics of Agents. In: Workshop on Social Intelligence Design (SID) (2007)
18. Kistler, F., Wißner, M., André, E.: Level of Detail based Behavior Control for Virtual Characters. In: Proc. of the 10th Int. Conf. on Intelligent Virtual Agents. Springer, Heidelberg (2010)
19. 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)
20. Giang, T., Mooney, R., Peters, C., O’Sullivan, C.: ALOHA: Adaptive Level of Detail for Human Animation: Towards a new Framework. In: EUROGRAPHICS 2000 Short Paper Proceedings, pp. 71–77 (2000)