TALKING SOUNDSCAPES: AUTOMATIZING VOICE TRANSFORMATIONS FOR CROWD SIMULATION

Jordi Janer[1], Roland Geraerts[2], Wouter G. van Toll[2], and Jordi Bonada[1]

[1]*Music Technology Group, Universitat Pompeu Fabra, Barcelona*

[2]*Department of Information and Computing Sciences, Utrecht University, Utrecht*

Correspondence should be addressed to Jordi Janer (jordi.janer@upf.edu)

ABSTRACT

The addition of a crowd in a virtual environment, such as a game world, can make the environment more realistic. While researchers focused on the visual modeling and simulation of a crowd, its sound production has received less attention. We propose the generation of the sound of a crowd by retrieving a very small set of speech snippets from a user-contributed database, and transforming and layering voice recordings according to the character localization in the crowd simulation. Our proof-of-concept integrates state-of-the-art audio processing and crowd simulation algorithms. The novelty resides in exploring how we can create a flexible crowd sound from a reduced number of samples, whose acoustic characteristics (such as people density and dialogue activity) could be modeled in practice by means of pitch, timbre and time-scaling transformations.

1. INTRODUCTION

Crowd simulation in games is a typical example of dynamic content, properly visually modelled but with a rather limited sonic representation. Usually the sound content is created by looping a unique background sound effect recording with a reduced interaction with the game dynamics (e.g. by triggering samples). At the other end, in terms of sonic flexibility, we find that text-to-speech synthesis is still not widespread in games due to its lack of realism to generate emotional and credible dialogues. Although we foresee an improvement of emotional synthesis technologies in the coming years, today a realistic crowd relies on processing existing recordings.

1.1. Soundscapes in virtual environments

Audio is a crucial element in immersive virtual environments. Its principal role is the creation of a sound ambience or soundscape. During the last decade, several technologies emerged to provide a more realistic and interactive audio content. On the one hand, one can find commercial sound engine middleware for games (e.g. FMOD[1] , Wwise[2]), and, on the other hand, more research-oriented real-time synthesis techniques (e.g. physical models [3] and sound ambience textural synthesis [4], [5]).

It is worth remarking that the results presented in this paper could also be generated by means of the above mentioned audio middleware. However, the contribution is to automate the sound design process to reduce tedious manual intervention for sound sample search.

To add realism when recreating a sonic environment, we should first avoid the perception of listening to a static or repetitive loop. Second, the quality of generated sounds should ensure that listeners recognize the nature of the sound sources.

To fulfill the first constraint, we implement a dynamic system with a graph model for sequencing individual sound events. For fulfilling the second constraint, there exist various parametric synthesis approaches, such as physical models or modal synthesis, that achieve high-quality environmental sounds. Moreover, Text-to-Speech (TTS) synthesizers achieve today an almost realistic speech for generating speech signals. However, synthesizers still permit a limited range of emotions, and a limited variety of voice timbre, which are the main research challenges. In this work, we decided to work with sample-based concatenative synthesis, using real-world speech recordings retrieved from an online user-contributed repository (Freesound [12]). We argue that

[1]http://www.fmod.com

[2]http://www.audiokinetic.com

with a very small set of recordings we can generate the necessary variability to build a realistic sound of a crowd. In past projects on the sonification of environmental soundscapes [7], we employed a similar strategies with good user feedback.

1.2. Realism in Crowd simulation

The research field of crowd simulation only exists for about 10 years. The basis unit is an agent representing a moving entity such as a human, a vehicle or a monster. A crowd can then be considered as a complex system that emerges from the agents' micro interactions where the agents move according to their internal states and goals.

Crowd simulation brings together techniques developed in different research areas, including path planning, collision avoidance, AI and social behavior, animation and visualization.

The first step to efficiently simulate a crowd is to create a navigation graph (such as a grid or waypoint graph [18]) or navigation mesh [16, 17] based on the walkable parts of the scene. This representation can then be queried by a shortest path algorithm such as A* [19]. The advantage of using a navigation mesh is that its 2D representation leads to more possibilities for the agents using the mesh, allowing them to avoid collisions with each other in a visually convincing way.

The first collision-avoidance models aimed at steering vehicles [20]. These models were simple but looked weird when applied to humans. More recently, a model was developed that predicts future collisions to mimic human behavior [21]. In this model, humans adapt their speed and direction in advance to avoid collisions with others. A related technique [22] uses Reciprocal Velocity Obstacles (RVOs) to compute the predicted free space for each human in real time. It was shown that predicting future routes leads to more natural paths than using RVOs [21]. However, both techniques can simulate thousands of agents in real-time.

On a global level, an agent must perform certain tasks such as 'try to capture the player's avatar'. To carry out this high-level task, a collection of actions needs to be computed. While scripts or Finite State Machines [18] are often used, they can become rather complex in current games because the number of agents and their interactions quickly grows. Since optimally solving this problem is NP-complete, heuristic such as Goal-Oriented Action Planning [23] are useful.

General crowd simulation systems have also been developed. Examples include the HiDAC model [24], composite agents [25], density-based crowd simulation [26] and scalable avoidance and group behaviors [27]. Animating and visualizing a large numbers of characters is an expensive part of the simulation. We refer the reader to [28, 29] for a good overview. These systems can steer hundreds of characters in real-time.

As the field maturates, methodologies for validating crowd simulations [30, 31] are surging.

1.3. Efficient audio rendering

One of the problems to be addressed in the context of computer games is the mixing of a large number of sound sources in real-time. Acoustic sources can represent the characters of a crowd, including vehicles and objects. Moreover, the allocation of audio engine processing time is limited to about 10% of CPU time in typical game software. Therefore, efficiency in mixing multiple audio tracks is a topic of interest in game audio.

For example, in [9], the authors describe an implementation of Stochastic Sampling. They present the use-case of a football stadium scene with more than $15,000$ football fans, yelling and singing. In the experiments, they tested sound rendering with sample sets of 20, 50, 150 and 2000 sound sources. As a conclusion, they argued that the rendering quality for 20 and 50 sources was not satisfactory. With 150 sources, they reached a quite plausible result for computer games using about 16% of CPU time.

Other works focus on the mixing limitations of multiple sources, which is principally a situation in interactive video games. In [10], the authors present a perceptually-based and scalable approach for efficiently filtering and mixing a large number of audio signals. They claim to yield a 3-fold to 15-fold improvement in the overall processing rate compared to brute-force techniques, with minimal degradation of the output.

Finally, yet in another context, auth [6] addresses the synthesis and usage of clapping sounds. Presenting some techniques and ideas used in that specific context can also be useful in the context of creating a more complex crowd simulation.

2. METHODOLOGY

2.1. Defining soundscape regions

In the presented talking soundscape, we define two types of sound content: *near-field* and *diffuse-field*. Near-field content consists of voices from individual agents that are at a short distance from the camera-view, and thus its spatialization (positions and direction of movement) is important. Voice sources should be uniquely distinguishable (timbre, prosody and timing). However, the speech signal does not need to be necessarily intelligible, since the crowd sound is not understood here as a part of the game dialogue. In diffuse-field content, we represent the crowd in a coarser manner by means of layering and sequencing mixtures of voice recordings. In real-time, the sound could be controlled only at a higher level, namely by mixing gains, speed and pitch-shifting. In both cases (i.e. near-field and diffuse-field), the synthesis is sampling-based and the sequencing of samples follows from a graph model. We used different configurations of graph models, as we will further explain in Section 2.4.

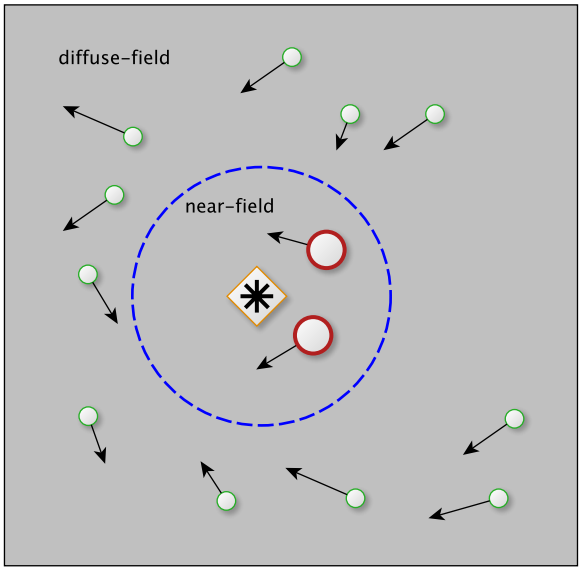

Fig. 1: Two acoustic regions are considered: diffuse-field and near-field. Sound sources are individually spatialized (distance and position) in the near-field region only.

2.2. Retrieval of speech samples

Most of the crowd sounds we find in publicly available Sound FX libraries correspond to specific situations such as sports events (e.g. 'wooing' in stadiums or arenas). To create a sound of a crowd in a pedestrian area or a shopping mall, the sonic result must be subtle yet realistic. Our approach departs from a small set of isolated speech or dialogue recordings. Moreover, we can retrieve these samples from a user-contributed repository (www.freesound.org), which allows diversity in the source audio content before the transformations.

Depending on the requirements of the crowd simulation in the near-field and diffuse-field population, we determine a number (S) of necessary different speech instances for each area. A large number of instances can be obtained by means of two strategies:

- *a)* large-scale retrieval: we search a large number S of different sounds in the Freesound.org repository, using text queries (e.g. 'speech', 'dialogue').
- *b)* small-scale retrieval plus transformation: we only use a reduced number of sounds (Q), and generate the necessary variation using L voice transformation presets to reach the total number of sample combinations $S = Q \cdot L$.

One goal of this work is to study how case *b)* can reduce the original speech sample dataset without compromising the realism of the crowd sound generation.

2.3. Transformation of speech samples

Once we have retrieved the speech samples, the next step is to apply voice transformations for obtaining a large variability of samples. Beyond phonetics, a speech signal is characterized by its timbre, prosody and timing. To create multiple realistic voices from a reduced set of recordings, we use a state-of-the-art method for real-time voice transformation [1, 2]. On the one hand, this algorithm allows applying pitch transposition in a range of two octaves without noticeable artifacts. On the other hand, it allows modifying the timbre of the input voice, e.g. by changing the gender (male to female) or the age (adult to child). When dealing with crowd simulation, such tools allow us to dynamically change the typology of the virtual agents in the crowd (e.g. a schoolyard).

As shown in Figure 2, we can generate multiple transformation output signals from a single speech recording. Moreover, it is worth to stress that transformation presets are stored as text files that can be generated automatically by means of scripting. Therefore, we can define the

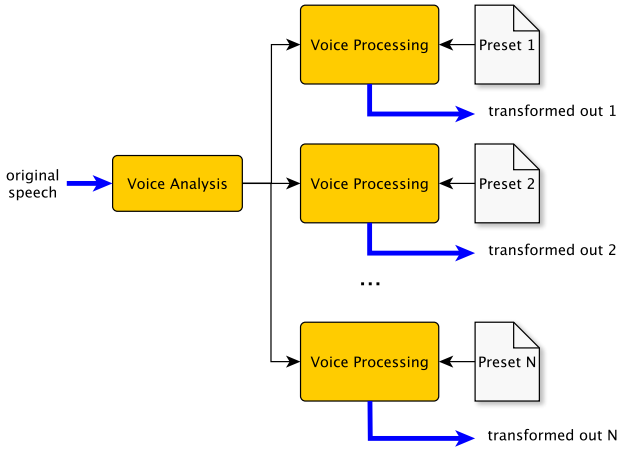

Fig. 2: Transformation presets can be generated automatically, producing several transformation files from a single speech input.

number of necessary transformations and automatically create all transformation presets.

Finally, although a real-time implementation is beyond the proof-of-concept presented in this article, the algorithms can generate up to 15 different simultaneous transformed voices in real-time, running on an off-the-shelf computer.

2.4. Sound Sequencing strategy

In the literature we find works that propose stochastic sampling methods [9]. In our case, we follow a different strategy.

The triggering of samples is driven by a graph model. Each node represents a single speech sample. When the sample playback is finished, an edge to the new node is selected, and the corresponding sample is played back. We use different strategies for diffuse-field and near-field:

- Diffuse-field: there is a single graph-model, with M_{DF} nodes and K agents traveling in the graph simultaneously. We define two functions to compute $K = f(d)$ and $M_{DF} = g(d)$ depending on a input parameter d (density). The spatialization here consists only in assigning a random panorama (left-right) value to each node in the graph model. Additionally, a reverb effect is added, applying a low-pass filter to simulate the frequency-dependent attenuation of acoustic wave propagation. The voice character C is a percentage, which specifies the ratio of nodes of one gender and age.

- Near-field: there are N graph models, one for each character in the scene. Each graph model is characterized individually in genre and age, where all M_{NF} samples in the graph model come from the same speaker or recording. M_{NF} is a configurable parameter. A single agent navigates in each graph model. Spatialization is stereo-based, where a monophonic speech signal is panned to left/right according to the angle with respect to the listener position and orientation. Gain is also updated in relation to the distance to the listener position.

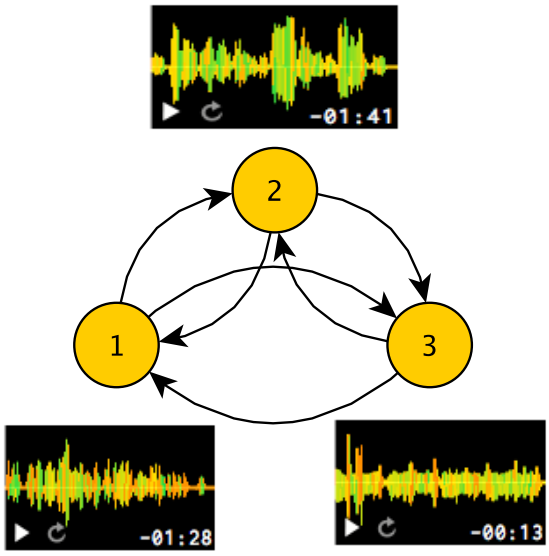

Fig. 3: Graph models used for sequencing the diffuse-field (one graph model with several samples of various speakers).

3. IMPLEMENTATION

We implemented a simulation as a proof-of-concept. The implementation consists of three independent modules: one for crowd simulation, one for graphics rendering and one for audio rendering. The rendering of the crowd sim-

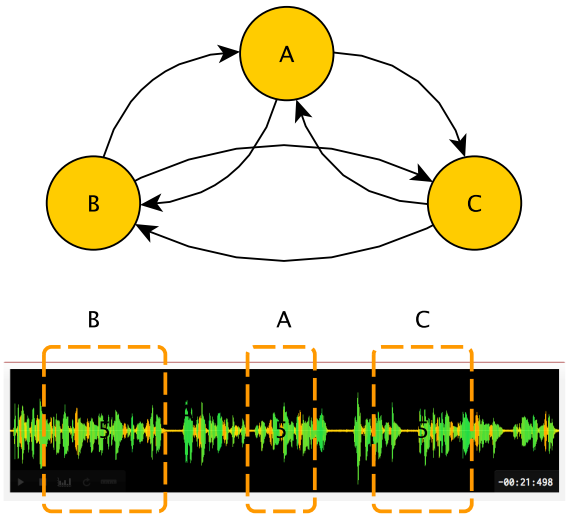

Fig. 4: Graph models used for sequencing the near-field (one graph model with several excerpts of the same speaker's sample).

ulation is done off-line. The path planning module generates a data structure containing information about the characters' positions and velocities, which are updated at a rate of 10 frames per second. The data structure is passed on to the two rendering modules for graphics and audio. The chosen scenario is a urban crossing. It is a common scene in computer games, featuring a dynamic human crowd moving in all directions.

3.1. Crowd simulation and graphics rendering

We generated a crowd in the environment shown in Figure 5. For this 3D environment, we have generated an obstacle representation in 2D, and a corresponding Explicit Corridor Map navigation mesh [13]. The environment measures 250 by 250 meters. As shown in Figure 6, we defined eight regions at the ends of the environment's sidewalks. Every 0.2 seconds, we created a character with its start and goal positions sampled in two randomly chosen (and different) regions. Hence, each character used the sidewalks and zebra crossings to move from one end of the environment to another. Our characters used the Indicative Route Method [14] and velocity-based collision avoidance [15] to smoothly move to their goals. To spread the crowd realistically, we assigned to each character a random preference for a certain side of the pavement, between -0.3 (left from the middle) and 0.8 (almost on the right). Each character had a maximum walking speed between 1.3 and 1.5m/s. With these settings, the crowd contained 900 characters on average.

Our crowd was simulated at 10 frames per second, which is common and leads to visually convincing results [14]. Naturally, the framerate of the encompassing application can be higher. We rendered the crowd in 3D from a fixed camera position. In every simulation step, we wrote the positions and velocities of each character to a file. This file served as the input for the audio engine.

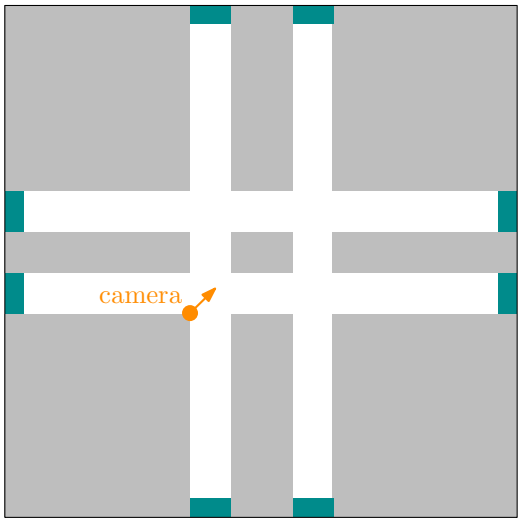

Fig. 6: A two-dimensional representation of our environment, used for character navigation. The walkable areas (sidewalks and zebra crossings) are shown in white. The rectangles at the sides indicate the regions from which we sampled the start and goal locations of each character. The position and direction of the camera for 3D visualization is shown in orange.

3.2. Sound rendering

We follow the methodology presented above in Section 2 for the retrieval, transformation and the sequencing of speech samples based on graph models for the near and diffuse fields. As an example, we selected 6 samples from the Freesound repository, querying with tags: "poetry+male" and "poetry+female". We manually chose three samples for each gender form the returned results respectively. By using the query "poetry", we enforced the retrieval of speech only samples with a calm recitative character.

Fig. 5: A screenshot of the visualized crowd in a 3D environment.

In this simulation, we defined as *near-field* the area around the listener position in a distance of 20 meters. For the near-field sequencing, each graph model consists of $M_{NF} = 10$ nodes, where each node corresponds to a sample excerpt of a random duration. All nodes are connected. At each time frame, the number of characters in the near-field area determines the number of created graph models. As mentioned before, a single agent navigates in each graph model.

In the near-field circle, we sequenced the audio at 10 frames per second, in correspondence with the crowd simulation. Outside this circle (i.e. in the *diffuse-field* region), we updated the number of agents K in our audio model every 5 seconds, since the number of simultaneous characters is approximated.

4. CONCLUSIONS

Currently, professional software for game audio allows designers to generate and sequence realistic sonic environments from audio samples or synthesis algorithms. However, in crowd simulation sounds, typical solutions have been to use single recordings of a crowd to avoid hours spent on sound design, finding speech samples and then programming complex audio sequencing. This approach presents ways for automatizing the sound design process, while voice transformation offers at the same time the necessary flexibility to change the characteristics of the crowd. We demonstrate this proof-of-concept in a short movie available online [3].

5. REFERENCES

[1] Bonada, J., "Voice Solo to Unison Choir Transformation", AES 118th Convention, Barcelona, 2005.

[2] Mayor, O. and Bonada, J. and Janer, J., "Audio transformation technologies applied to video games", 41st AES Conference: Audio for Games, London, 2011.

[3] A. Farnell, "Designing Sound - Practical synthetic sound design for film, games and interactive media using dataflow". London: Applied Scientific Press Ltd., 2008.

[4] S. Kersten and H. Purwins, "Sound texture synthe- sis with hidden markov tree models in the wavelet domain, in The 7th Sound and Music Computing Conference, Barcelona, 2010.

[3] http://www.dtic.upf.edu/ jjaner/presentations/aes2013

[5] D. Schwarz and S. Norbert, “Descriptor-based sound texture sampling, in The 7th Sound and Mu- sic Computing Conference, Barcelona, 2010.

[6] L. Peltola, C. Erkut, P. R. Cook, and V. VŁlimŁki, “Synthesis of hand clapping sounds,” IEEE Transactions on Audio, Speech and Language Processing, vol. 15, pp. 1021-1029, March 2007.

[7] Janer, J., Kersten S., Schirosa M., and Roma G. (2011). An online platform for interactive soundscapes with user-contributed content. AES 41st International Conference on Audio for Games. 2011

[8] Akker, J.M. van den, Geraerts, R.J., Hoogeveen, J.A. and Prins, C.J., “Path Planning in Games”. In 10th Workshop on Models and Algorithms for Planning and Scheduling Problems. Nymburk, Czech Republic. 2011.

[9] Wand, M., Strasser, W., “A Real-Time Sound Rendering Algorithm for Complex Scenes”, Technical Report, University of Tübingen, 2003.

[10] Tsingos, N., “Scalable perceptual mixing and filtering of audio signals using an augmented spectral representation”, Proc. of the 8th Int. Conference on Digital Audio Effects, Madrid, Spain, September 20-22, 2005.

[11] Raghuvanshi, N. et al., “Real-Time Sound Synthesis and Propagation for Games”, Communications of the ACM, Vol. 50, No. 7 67–73, 2007.

[12] Freesound.org, Universitat Pompeu Fabra, `http://www.freesound.org`, 2005.

[13] Geraerts, R., “Planning short paths with clearance using Explicit Corridors”, In IEEE International Conference on Robotics and Automation, pp 1997–2004, 2010

[14] Karamouzas, I. and Geraerts, R. and Overmars, M.H., “Indicative routes for path planning and crowd simulation”, In International Conference on Foundations of Digital Games, pp 113–120, 2009.

[15] Karamouzas, I. and Overmars, M.H., “A velocity-based approach for simulating human collision avoidance”, Lecture Notes in Computer Science, Intelligent Virtual Agents, Springer, pp 180–186, vol 6356, 2010.

[16] Pettre, J., Laumond, J.-P. and Thalmann, D., “A navigation graph for real-time crowd animation on multilayered and uneven terrain.” In The First International Workshop on Crowd Simulation, 2005.

[17] van Toll, W.G., Cook IV, A.F. and Geraerts, R., “Navigation Meshes for Realistic Multi-Layered Environments”, In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3526-3532, 2011.

[18] Rabin, S., “AI Game Programming Wisdom 4”. Charles River Media, 2008.

[19] Hart, P., Nilsson, N. and B. Raphael, ”A Formal Basis for the Heuristic Determination of Minimum Cost Paths”, IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100-107, 1968.

[20] Reynolds, C. W., “Flocks, herds and schools: A distributed behavioral model”, SIGGRAPH Comput. Graph., vol. 21, pp. 25–34, 1987.

[21] Karamouzas I., Heil P., van Beek P. and Overmars M.H., “A predictive collision avoidance model for pedestrian simulation”, In 2nd International Workshop on Motion in Games, Zeist, The Netherlands, pp 41-52, 2009.

[22] van den Berg J.P., Lin M.C and Manocha D. “Reciprocal Velocity Obstacles for real-time multi-agent navigation”, In IEEE International Conference on Robotics and Automation, Pasadena, California, USA, pp. 19281935, 2008.

[23] Orkin J. “Three states and a plan: The AI of F.E.A.R.”, In Game Developers Conference, 2006.

[24] Pelechano, N., Allbeck J.M. and Badler, N. I., “Controlling individual agents in high-density crowd simulation”, In Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 99-108, 2007.

[25] Yeh, H., Curtis, S., Patil, S., van den Berg, J., Manocha, D., and Lin, M., “Composite agents”, In Eurographics/ACM SIGGRAPH Symposium on Computer Animation, pp. 39-47, 2008.

[26] van Toll, W.G., Cook IV, A.F. and Geraerts, R., “Real-Time Density-Based Crowd Simulation”, Computer Animation and Virtual Worlds, 23(1):59-69, 2012.

[27] Yersin, B., Maïm, J., Morini, F. and Thalmann, D., “Real-time crowd motion planning: Scalable Avoidance and Group Behavior”, Vis Comput 24: 859870, 2008.

[28] Ryder, G. and Day, A.M., “Survey of Real-Time Rendering Techniques for Crowds”, Computer Graphics forum, 24(2):203-215, 2005.

[29] Tecchia, F., Loscos, C. and Chrysanthou, Y., “Visualizing Crowds in Real-Time”, Computer Graphics forum,21(4):753-765, 2002

[30] Lerner, A., Chrysanthou, Y., Shamir, A. and Cohen-Or, D., “Data Driven Evaluation of Crowds”, In Motion in Games, Springer Lecture Notes in Computer Science (LNCS) 5884, pp. 75–83, 2009.

[31] Singh, S., Kapadia, M., Faloutsos, P. and Reinman, G., “SteerBench: a benchmark suite for evaluating steering behaviors”, Computer Animation and Virtual Worlds, 20:533–548, 2009.